Diss. ETH No. 16365

On Graphical Models for Communications and Machine Learning: Algorithms, Bounds, and Analog Implementation

A dissertation submitted to the
Swiss Federal Institute of Technology, Zürich
for the degree of
Doctor of Sciences ETH Zürich

presented by

Justin H. G. Dauwels

ir.
born on November 2, 1977
citizen of Belgium

accepted on the recommendation of
Prof. Dr. Hans-Andrea Loeliger, examiner
Prof. Dr. Shun-ichi Amari, co-examiner
Prof. Dr. Marc Moeneclaey, co-examiner
Dr. Jonathan Yedidia, co-examiner

Hartung-Gorre Verlag, Konstanz, May 2006

Series in Signal and Information Processing **Vol. 17**

Editor: Hans-Andrea Loeliger

Bibliographic Information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the internet at http://dnb.ddb.de.

First Edition 2006

HARTUNG-GORRE VERLAG KONSTANZ

ISSN 1616-671X
ISBN 3-86628-080-7

Man muss noch Chaos in sich haben
um einen tanzenden Stern gebären zu können.
(F. Nietzsche)

Science is the belief in the ignorance of experts.
(R. P. Feynman)

Acknowledgments

Above all, I would like to express my deepest gratitude to my "Doktorvater" Andi Loeliger. It's hard not to get infected by Andi's extraordinary drive and passion for research and his good taste for research problems. Andi was a great advisor to me, but I'm sure I will never attain his level of clear thinking. I also wish to thank Andi for giving me the opportunity to attend numerous international conferences and workshops and to visit other research groups.

I'm strongly indebted to Jonathan Yedidia for the wonderful time I had as an intern at MERL under his guidance. I really enjoyed our numerous discussions at the white board. I also thank Jonathan for sharing with me his deep insight in the connections between inference and statistical physics.

Many thanks go to Marc Moeneclaey for giving me several times the chance to visit his research group. The generous and competent feedback from Marc and his co-workers has strongly influenced this thesis. In particular, I would like to thank Henk Wymeersch for sharing with me his ideas on phase synchronization and channel estimation in general, and for taking me out to the better restaurants in Gent.

I'm very grateful to Shun-ichi Amari for having agreed to serve as co-examiner for this doctoral thesis, and for accepting me as a post-doctoral researcher in this research lab. Domo arigato gozaimashita.

I would like to thank Neil Gershenfeld for arranging my stay at MIT, introducing me to his extraordinary research group, and for his warm hospitality.

It was also a great pleasure to share my office with several visitors of our lab, in particular, Li Ping, Mehdi Molkaraie, and Frank Kschischang. I'm grateful to Li Ping for patiently explaining to me his inspiring and truly innovative ideas. I thank Mehdi Molkaraie for the many interesting discussions on statistical physics, and for feeding me with Swiss chocolate every time I was about to run out of energy. I'm grateful to Frank Kschischang for his generous advice, his continued support and encouragement, and his great sense of humor.

I wish to thank Hagit Messer-Yaron for her constructive comments on this thesis and for pointing out some important references, and Dave Forney for his encouraging and inspiring feedback on our projects related to factor graphs.

This thesis strongly benefited from the generosity and wit of my friends and colleagues. I'm grateful to Sascha Korl for the numerous fruitful coffee-break discussions, where many ideas of this thesis initiated (and many others ended in the garbage bin). Special thanks go to Volker Koch for persistently reminding me of the wise words "In der Ruhe liegt die Kraft!"—Volker, I will get the point one day— and for the pleasant time we spent together in and outside the lab. I would like to thank Ralf Kretzschmar for guiding me through the culinary life of Zurich and for our interesting discussions on the theory and practice of neural networks, Markus Hofbauer for continuously updating me with the hippest salsa dance steps and grooviest bongo rythms, and all other members of the Signal and Information Processing Laboratory (ISI) at ETH for being great colleagues, in particular, Dieter Arnold, Jonas Biveroni, Murti Devarakonda, Matthias Frey, Qun Gao, Daniel Hoesli, Junli Hu, Tobias Koch, Patrick Merkli, Natalia Miliou, Stefan Moser, Maja Ostojic, Stephan Tinguely, Pascal Vontobel, Michèle Wigger, and Daniel Zennaro.

I also would like to thank the staff of the ISI, in particular, Max Duenki and Patrick Strebel for keeping the UNIX system up and running smoothly, Bernadette Roeoesli for organizing my traveling, Thomas Schaerer for patiently and competently answering my silly questions on electronics, and Francesco Amatore for his kind support in arranging my office.

I'm grateful to my parents for their support and love throughout my life; without them, none of this would have been possible.

At last, a big kiss to my wife Shoko for too many reasons to list here.

Abstract

This dissertation is about a specific problem and about general methods. The specific problem is carrier-phase synchronization, which appears in the context of digital communications. The general methods are message-passing algorithms operating on graphical models, in particular, factor graphs. We consider applications of such algorithms in the context of statistical inference (as in communications, signal processing, and machine learning), statistics, information theory, and the theory of dynamical systems (such as analog electronic circuits).

The primary motivation for this work was (1) to analyze the degradation of digital communications systems due to oscillator non-idealities; (2) the development of synchronization algorithms that minimize this performance degradation.

Clocks are ubiquitous in digital communications systems; real-life clocks are noisy, i.e., their signals are not perfectly periodic, which often leads to a significant degradation of the performance of communications systems.

In the early days of communications, this source of degradation was only of secondary concern. Communications systems used to operate far from the ultimate performance bound, i.e., channel capacity. The main concern was therefore to develop error-correcting techniques that could close the gap between the performance of practical communications systems and channel capacity.

With the recent advent of iterative decoding techniques, communications systems nowadays most often operate close to the ultimate performance limits; issues such as synchronization, which were earlier only of secondary importance, have now become the mayor (remaining) bottlenecks in the design of communications systems.

In this dissertation, we focus on carrier-phase synchronization, i.e., the alignment of the phase of the local oscillator in the receiver to the phase of the incoming carrier. The questions we address are:

a) Which physical mechanisms are responsible for phase noise? How can phase noise be modeled?

b) How can carrier-phase estimation algorithms systematically be derived?

c) What are the ultimate limits for communication over channels with phase noise? In particular:

    i) How much does the information rate of a communications channel decrease due to phase noise?

    ii) How well can the (noisy) carrier phase be estimated?

In contrast to earlier and parallel work, our aim is not the design and optimization of fully operating communications systems. In this thesis, various tools are developed that lead (or may lead) to an answer to the above questions (and many other related questions).

We give a detailed analysis of phase noise in free-running clocks and PLLs (Question 1). We propose a simple intuitive model for phase noise in free-running oscillators. We describe two simple models for passband communications channels. The models take phase offsets into account between the received carrier and the local carrier in the receiver, but disregard timing offsets. In the first model, the phase is constant, in the second, the phase performs a random walk. We investigate under which conditions the two models are valid. Most methods of this thesis will be illustrated by means of both channel models.

Most methods we propose in this dissertation are based on graphical models, more precisely, factor graphs. Factor graphs are used to visualize the structure of the system at hand. They represent the factorization of multivariate functions. Each edge in the graph corresponds to a variable, each node corresponds to a factor. Factor graphs can represent any function, in particular, probabilistic models, error-correcting codes, block diagrams and other common models in communications, signal processing and beyond.

We show how factor graphs can be used as a tool to develop practical estimation and detection algorithms. Our techniques can be applied to model-based signal processing (e.g., phase estimation) and machine learning. In particular, we formulate several standard signal-processing and machine-learning algorithms as message passing on factor graphs, e.g., particle methods, gradient methods, decision-based methods, and expectation maximization. In all those algorithms, local rules are applied at the nodes in a factor graph. In other words, the (global) estimation and detection problem is tackled by a divide-and-conquer strategy: the global computation is carried out by multiple (simple) local computations. The local message-update rules may be used as building blocks for novel estimation and detection algorithms. By listing the possible update rules at each node in the factor graph, one can systematically explore novel algorithms. We demonstrate this idea by deriving phase estimation algorithms for the constant-phase model and the random-walk phase model (Question 2). We also show how the back-propagation algorithm for the training of feed-forward neural networks follows by applying generic message-passing rules. We elaborate on the computation of kernels in the light of message passing on factor graphs.

We demonstrate how message-passing algorithms for inference can be implemented as dynamical systems, in particular, as clock-free analog electronic circuits. Those systems operate in continuous time, and do not require a digital clock; therefore, they circumvent the problem of timing synchronization.

We present a numerical algorithm to compute the information rate of continuous channels with memory (Question 3.a). The algorithm is an extension of the methods proposed earlier for discrete channels with memory. Also here, factor graphs and the summary-propagation algorithm are key ingredients. We apply the method to the random-walk phase model. The algorithms we propose for computing Cramér-Rao-type bounds open the door to exciting applications of information geometry, such as (1) natural-gradient-based algorithms; (2) the computation of Fisher kernels.

We propose a numerical algorithm for computing the capacity (or lower bounds on capacity) of continuous memoryless channels (Question 3.a). We present numerical results for the Gaussian channel with average-power and/or peak-power constraints. We outline how the algorithm can be extended to continuous channels with memory (e.g., channels with phase noise) by means of message-passing techniques.

We propose message-passing algorithms to compute Cramér-Rao-type bounds. Cramér-Rao-type bounds are lower bounds on the minimum mean square estimation error; the bounds may be used to asses the performance of practical (message-passing) estimation algorithms, in particular, our phase-estimation algorithms (Question 3.b). The algorithms we propose for computing Cramér-Rao-type bounds open the door to exciting applications of information geometry, such as (1) natural-gradient-based algorithms; (2) the computation of Fisher kernels.

Keywords: graphical models, summary-propagation, belief propagation, message passing, expectation maximization, EM, steepest descent, particle filter, MCMC, particle methods, Gibbs sampling, importance sampling, decision-based estimation, iterative conditional modes, ICM, carrier phase estimation, phase noise, clock jitter, synchronization, Blahut-Arimoto algorithm, information rate, channel capacity, Cramér-Rao bound, information matrix, kernel methods, Fisher kernel, product kernel, probabilistic kernel, neural networks, back-propagation algorithm, analog electrical circuits, linear feedback shift register, LFSR.

Kurzfassung

Diese Dissertation beschreibt einerseits ein spezifisches Problem und andererseits allgemeine Methoden. Das spezifische Problem ist Trägerphasensynchronisation, welches im Kontext der digitalen Kommunikation auftritt. Die allgemeinen Methoden sind sogenannte "message-passing" Algorithmen, welche auf graphischen Modellen angewandt werden, insbesondere auf Faktorgraphen. Wir betrachten Anwendungen solcher Algorithmen im Kontext von statistischer Inferenz (wie z.B. in der digitalen Kommunikation, in der Signalverarbeitung oder im maschinellen Lernen), Statistik, Informationstheorie und der Theorie dynamischer Systeme (wie z.B. analoge elektrische Schaltungen).

Die primäre Motivation für diese Arbeit war (1) den Leistungsverlust digitaler kommunikationssysteme infolge nicht-idealen Oszillatoren zu untersuchen; (2) die Entwicklung von Algorithmen, welche diesen Leistungsverlust minimieren.

Oszillatoren sind allgegenwärtig in digitalen Kommunikationssystemen; Signale praktischer Oszillatoren sind verrauscht, d.h., die Oszillatorsignale sind nicht exakt periodisch, was oft zu einem bedeutenden Leistungsverlust der Kommunikationssysteme führt.

In den frühen Tagen der Kommunikation war diese Art von Leistungsverlust eher von sekundärer Bedeutung. Die Kommunitionssysteme operierten zu diesen Zeiten weit entfernt von der theoretischen Leistungsgrenze, d.h., die Kanalkapazität. Die Hauptsache war deswegen Fehlerkorrigierende Methoden zu entwickeln, welche die Lücke zwischen der Leistung praktischer Kommunikationssysteme einerseits und der Kanalkapazität andererseits schliessen könnten.

Mit der jüngsten Entwicklung iterativer Dekodierungsmethoden arbeiten die heutigen Kommunikationsysteme nahe an der theoretischen Leistungsgrenze; Probleme wie Synchronisation, welche zuvor eher von sekundärer Bedeutung waren, sind heute die wichtigsten (letzten) Engpässe beim Entwurf von Kommunikationssystemen geworden.

In dieser Dissertation konzentrieren wir uns auf Trägerphasensynchronisation. Darunter versteht man das Synchronisieren der Phase des lokalen Empfängersoszillators mit der Phase des empfangenen Signals.

Die Fragen welche wir betrachten sind:

a) Welche physikalische Mechanismen sind verantwortlich für Phaserauschen? Wie kann Phaserauschen modelliert werden?

b) Wie können Trägerphasenschätzungsalgorithmen systematisch hergeleitet werden?

c) Was sind die theoretischen Leistungsgrenzen für Kommunikation über Kanäle mit Phaserauschen, insbesondere:

    i) Wieviel nehmen die Informationsraten infolge Phaserauschen ab?

    ii) Wie gut kann die (verrauschte) Tragerphase geschätzt werden?

Im Gegensatz zu früheren und parallelen Arbeiten ist unser Ziel nicht der Entwurf und die Optimierung von vollfunktionierenden Kommunikationssystemen. In dieser Arbeit werden verschiedene Werkzeuge entwickelt, welche zu Antworten auf die obenstehenden Fragen (und vielen anderen verwandten Fragen) führen (oder führen könnten).

Wir beschreiben eine detaillierte Analyse von Phasenrauschen in freilaufenden Oszillatoren und PLLs (Frage 1). Wir schlagen ein einfaches intuitives Modell für Phaserauschen in freilaufenden Oszillatoren vor. Wir beschreiben zwei einfache Modelle für Bandpass-Kommunikationskanäle. Die Modelle berücksichtigen Offsets zwischen der Phase des Empfangersoszillators und der Phase des empfangenen Signals; die Modelle vernachlässigen aber Timing-Offsets. Im ersten Modell ist der Phase-Offset konstant, im zweiten Modell wird der Phase-Offset modelliert als ein Random-Walk Prozess. Wir untersuchen unter welchen Bedingungen beide Modelle gültig sind. Die meisten Methoden dieser Dissertation werden anhand beiden Modellen illustriert.

Viele (wenn nicht alle) Methoden dieser Arbeit sind basiert auf graphischen Modellen, insbesondere Faktorgraphen. Faktorgraphen werden u.a. verwendet um die Struktur des betrachteten Systems zu visualisieren. Sie stellen die Faktorisierung multivariater Funktionen dar. Jede Kante des Graphens entspricht einer Variable, jeder Knoten des Graphens entspricht einem Faktor. Faktorgraphen können beliebige Funktionen darstellen, insbesondere probabilistische Modelle, fehlerkorrigierende Codes, Blockdiagramme und andere Modelle welche oft in der Kommunikation, in der Signalverarbeitung und in anderen Gebieten verwendet werden.

Wir zeigen wie Faktorgraphen als ein Werkzeug verwendet werden können, um praktische Schätz- und Detektionsalgorithmen zu entwickeln. Unsere Techniken können auf modellbasierte Signalverarbeitung (z.B. Phaseschätzung) und auf machinelles Lernen angewandt werden. Wir formulieren verschiedene Standard-algorithmen der Signalverarbeitung und des machinellen Lernens als message passing auf Faktorgraphen, z.B. Partikelmethoden, Gradientenverfahren, entscheidungsbasierte Methoden, und expectation maximization (EM). In diesen Algorithmen werden lokale Regeln an den Knoten des Faktorgraphen angewandt. Mit anderen Worten, die globale Schätzaufgabe (oder Detektionsaufgabe) wird anhand einer teile-und-herrsche Strategie angepackt: die globale Berechnung wird durch viele (einfache) lokale Berechnungen ersetzt. Die lokale Regeln können als Bausteine für neue Schätz- und Detektionsalgorithmen verwendet werden. Durch dem Auflisten von den möglichen Aufdatierungsregeln an jedem Knoten des Faktorgraphen kann man systematisch neue Algorithmen entwicklen. Wir demonstrieren diese Idee für das konstante-Phasemodell und das Random-Walk Phasemodell (Frage 2). Wir zeigen auch wie der back-propagation Algorithmus für das Trainieren von feed-forward neuronalen Netzwerken als das Anwenden von generischen Aufdatierungsregeln auf einem geeigneten Faktographen aufgefasst werden kann. Wir beschreiben Kernels im Kontext von message passing auf Faktorgraphen.

Wir zeigen wie message-passing Algorithmen für Inferenz als dynamische Systeme implementiert werden können, insbesondere als Uhr-freie analoge elektrische Schaltungen. Diese Systeme arbeiten zeitkontinuierlich and brauchen deswegen keine digitale Uhr; deshalb vermeiden sie das

Problem von Zeitsynchronisation.

Wir schlagen eine nummerische Methode vor, um Informationsraten von kontinuierlichen Kanälen zu berechnen (Frage 3.a.). Der Algorithmus ist eine Erweiterung einer Methode welche zuvor für diskrete Kanäle vorgeschlagen wurde. Auch hier sind Faktorgraphen zusammen mit dem summary-propagation Algorithmus ein wichtiger Bestandteil. Wir wenden die Methode auf das random-walk Phasemodell an.

Eine nummerische Methode für die Berechnung von Kapazitäten (oder den unteren Grenzen für die Kapazität) von kontinuierlichen Kanälen ohne Gedächtnis (Frage 3.a) wird vorgeschlagen. Wir bieten nummerische Ergebnisse für den Gauss'schen Kanal mit mittleren- und maximalen-Leistungsbedingungen an. Wir skizzieren wie der Algorithmus mit Hilfe von message-passing Methoden auf Kanäle mit Gedächtnis (z.B. Kanäle mit Phaserauschen) erweitert werden kann.

Wir schlagen message-passing Algorithmen für die Berechnung von Cramér-Rao-Typ Grenzen vor. Cramér-Rao-Typ Grenzen sind untere Grenzen für den minimalen mittleren quadratischen Schätzfehler; diese Grenzen können verwendet werden um praktische (message-passing) Schätzalgorithmen zu bewerten, insbesondere unsere Phaseschätzer (Frage 3.b). Die Algorithmen für die Berechnung von Cramér-Rao-Typ Grenzen welche wir vorschlagen könnten zu neuen interessanten Anwendungen der Informationsgeometrie führen, wie z.B. (1) natürlicher-gradienten-basierte Algorithmen; (2) die Berechnung von Fisher-Kernels.

Stichworte: Graphische Modelle, summary-propagation, belief propagation, message passing, expectation maximization, EM, steepest descent, Partikelfilter, MCMC, Partikelmethoden, Gibbs sampling, importance sampling, entscheidungsbasiertes Schätzen, iterative conditional modes, ICM, Trägerphasenschätzung, Phasenrauschen, clock jitter, Synchronisation, Blahut-Arimoto Algorithmus, Informationsrate, Kanalkapazität, Cramér-Rao Grenze, Informationsmatrize, Kernel-Methoden, Fisher-Kernel, Produkt-Kernel, probabilistischer Kernel, neuronale Netzwerke, back-propagation Algorithmus, analoge elektrische Schaltungen, LFSR.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

This dissertation is about:

- a *particular* problem, i.e., carrier-phase synchronization, which appears in the context of digital communications.
- *general methods*, i.e., message-passing algorithms operating on graphical models, in particular, factor graphs. We consider applications in the context of statistical inference (as in communications, signal processing, and machine learning), statistics, information theory, and the theory of dynamical systems (such as analog electronic circuits).

The primary motivation for this work was (1) to analyze the degradation of digital communications systems due to oscillator non-idealities; (2) the development of synchronization algorithms that minimize this performance degradation.

Clocks are ubiquitous in digital communications systems; real-life clocks are noisy, i.e., their signals are not perfectly periodic, which often leads to a significant degradation of the performance of communications systems.

In the early days of communications, this source of degradation was only of secondary concern. Communications systems used to operate far from the ultimate performance bound, i.e., channel capacity. The main concern was therefore to develop error-correcting techniques that could close the gap between the performance of practical communications systems and channel capacity.

With the recent advent of iterative decoding techniques, communications systems nowadays most often operate close to the ultimate performance limits; issues such as synchronization, which were earlier only of secondary importance, have now become the mayor (remaining) bottlenecks in the design of communications systems.

In this dissertation, we focus on carrier-phase synchronization, i.e., the alignment of the phase of the local oscillator in the receiver to the phase of the incoming carrier. The questions we address are:

a) Which physical mechanisms are responsible for phase noise? How can phase noise be modeled?

b) How can carrier-phase estimation algorithms systematically be derived?

c) What are the ultimate limits for communication over channels with phase noise? In particular:

    i) How much does the information rate of a communications channel decrease due to phase noise?

    ii) How well can the (noisy) carrier phase be estimated?

In contrast to earlier and parallel work, our aim is not the *design* and *optimization* of fully operating communications systems. In this dissertation, various *tools* are developed that lead (or may lead) to an answer to the above questions (and many other related questions).

Most of the methods we propose in this dissertation are based on graphical models, more precisely, factor graphs [119] [103] [66]. Factor graphs are used to visualize the structure of the system at hand. They represent the factorization of multivariate functions. Each edge in the graph corresponds to a variable, each node corresponds to a factor. Factor graphs can represent any function, in particular, probabilistic models, error-correcting codes, block diagrams and other common models in communications, signal processing and beyond.

Factor graphs can be used for statistical inference, i.e., detection and estimation. Statistical inference is performed by sending messages along the edges of the graph ("summary propagation" or "message passing"). Different algorithms are obtained by different message types or different message-update schedules. We will derive various phase-estimation algorithms within this framework (Question 2). We show how various existing algorithms can be interpreted as message passing on factor graphs, e.g., particle methods, decision-based methods, the backpropagation algorithm for the training of feedforward neural networks, etc.

We demonstrate how message-passing algorithms for inference can be implemented as *dynamical systems*, in particular, as clock-free analog electronic circuits. Those systems operate in continuous time, and do not require a digital clock; therefore, they circumvent the problem of timing synchronization.

A different application of factor graphs is the computation of information rates of (discrete) channels with memory (Question 3.a). As has been shown in [12], information rates for such channels can be computed by forward-only messaging on the graph of the state-space model that represents the channel. In this dissertation, we extend this result to continuous channels. Moreover, we investigate how the *capacity* of continuous channels can be computed.

In this dissertation, we will also present an entirely novel application of factor graphs: the computation of Cramér-Rao-type bounds, which are lower bounds on the mean squared estimation error. We will compute those bounds for a communications channel with phase noise (Question 3.b). The algorithms we propose for computing Cramér-Rao-type bounds open the door to exciting applications of information geometry, such as (1) natural-gradient-based algorithms; (2) the computation of Fisher kernels.

Many of the tools we propose in this dissertation are applicable to a much wider variety of problems than merely phase estimation, i.e., they are not only applicable to synchronization or other estimation/detection problems in digital communications, but also to problems in signal processing and machine learning.

1.2 How to read this thesis?

This dissertation may be of interest to:

- *communications engineers*, who want to learn more about (1) algorithmic and information-theoretic aspects of carrier-phase estimation or the problem of channel estimation in general; (2) the design of clock-free analog circuits for pseudo-noise synchronization, or, more generally, the implementation of message-passing algorithms by means of dynamical systems.

- researchers in *signal processing* and *machine learning*, who want to learn more about (1) message-passing algorithms for estimation; (2) the computation of performance bounds for estimation algorithms; (3) the connection between, on the one hand, message passing on graphical models and, on the other hand, neural networks, information geometry and kernel machines.

Each chapter of this thesis concerns a specific aspect of the problem at hand, and can, to a great extend, be read independently of the others. We provide an introduction to factor graphs and the sum(mary)-product algorithm in Chapter 2. We recommend the reader who is not familiar with factor graphs and the sum(mary)-product algorithm to start with Chapter 2 before reading any following chapter. In the next section, we outline the content of this thesis and mention our contributions.

1.3 Outline

Channel Model (Question 1)

We give a detailed analysis of phase noise in free-running clocks and PLLs. We propose a simple intuitive model for phase noise in free-running oscillators. The model is an alternative to the more sophisticated analysis by Demir et al. [54] [57] based on Floquet-theory.

We describe two simple models for passband communications channels. The models take phase offsets into account between the received carrier and the local carrier in the receiver, but disregard timing offsets. In the first model, the phase is constant, in the second, the phase performs a random walk. We investigate under which conditions the two models are valid. Most methods of this thesis will be illustrated by means of both channel models.

Factor Graphs and Summary Propagation

This chapter gives a brief introduction to factor graphs and the summary-propagation algorithm.

Phase Estimation Algorithms (Question 2)

We show how factor graphs can be used as a tool to develop practical estimation and detection algorithms. Our techniques can be applied to model-based signal processing (e.g., carrier-phase estimation) and machine learning.

In particular, we formulate several standard signal-processing and machine-learning algorithms as message passing on factor graphs, e.g., particle methods, gradient methods, decision-based methods, and expectation maximization. In all those algorithms, local rules are applied at the nodes in a factor graph. In other words, the (global) estimation and detection problem is tackled by a divide-and-conquer strategy: the global computation is carried out by multiple (simple) local computations. The local message-update rules may be used as building blocks for novel estimation and detection algorithms. By listing the possible update rules at each node in the factor graph, one can *systematically* explore novel algorithms.

We demonstrate this idea by deriving phase estimation algorithms for the constant-phase model and the random-walk phase model.

Appendix E and D are strongly based on the results of this chapter.

In Appendix E, we show how the back-propagation algorithm for the training of feed-forward neural networks follows by applying generic rules on a suitable factor graph.

In Appendix D, we investigate how kernels can be extracted from graphical models by means of message passing on factor graphs.

The message-passing tools presented in this chapter were developed in collaboration with Sascha Korl.

Computing Cramér-Rao-type Bounds (Question 3.b)

We propose message-passing algorithms to compute Cramér-Rao-type bounds. Cramér-Rao-type bounds are lower bounds on the minimum mean square estimation error; the bounds may be used to asses the performance of practical (message-passing) estimation algorithms, in particular, our phase-estimation algorithms. The algorithms we propose for computing Cramér-Rao-type bounds open the door to exciting applications of information geometry, such as (1) natural-gradient-based algorithms; (2) the computation of Fisher kernels.

We wish to acknowledge Shun-ichi Amari, Sascha Korl, Frank Kschischang, Amos Lapidoth, and Marc Moeneclaey for inspiring discussions and useful feedback on the topics of this chapter.

Computing Information Rates of Continuous Channels with Memory (Question 3.a)

We present a numerical algorithm to compute the information rate of *continuous* channels with memory (Question 3.a). The algorithm is an extension of the methods proposed earlier for *discrete* channels with memory [12] [179] [160]. Also here, factor graphs and the summary-propagation algorithm are key ingredients. We apply the method to the random-walk phase model.

Capacity of Continuous Memoryless Channels (Question 3.a)

A numerical algorithm is presented for computing the capacity (or lower bounds on capacity) of continuous memoryless channels. We present numerical results for the Gaussian channel with average-power and/or peak-power constraints. We outline how the algorithm can be extended to continuous channels with memory (e.g., channels with phase noise) by means of message-passing techniques.

We wish to acknowledge Sascha Korl and Frank Kschischang for inspiring discussions on the topic of this chapter.

Analog Clockless Electronic Circuit for PN-Synchronization

This chapter does *not* address one of the three questions we listed in the above. Its topic, however, is strongly related to (1) the problem of synchronization; (2) message passing on factor graphs.

We present an analog electronic circuit that synchronizes to pseudo-noise sequences. The circuit operates without a digital clock, and avoids therefore the problem of timing synchronization. We derive the circuit as message passing on a suitable factor graph. In this fashion, we established a connection between statistical state estimation and the phenomenon of entrainment.

The results presented in this chapter are based on joint work with Matthias Frey, Neil Gershenfeld, Tobias Koch, Patrick Merkli and Benjamin Vigoda. My personal contribution concerns the statistical estimation aspect, and not the hardware implementation or measurement of the circuit.

Conclusions and Outlook

The last chapter states some concluding remarks and suggestions for future research.

Appendices

In the appendices, we provide background information concerning estimation and detection theory (Appendix A), information theory (Appendix B), coding theory (Appendix C), kernel methods (Appendix D), neural networks (Appendix E), and Kalman filtering (Appendix H with related material in Appendix F and Appendix G). In Appendix I, we provide necessary conditions for differentiation under the integral sign, an operation we will often carry out in this thesis.

Appendix D also contains some thoughts on how kernels can be derived from graphical models, in particular, by message-passing methods.

In Appendix E, we in addition show how the back-propagation algorithm for the training of feed-forward neural networks can be derived as message-passing on factor graphs.

In the Appendices J–L, we provide detailed derivations of some results presented in this thesis.

Appendix J contains the derivations of the EM update rules listed in Section 4.9.2.

In Appendix K, we give proofs of lemmas and theorems stated in Chapter 5.

In Appendix L, we derive alternative update rules for the soft-LFSR presented in Section 8.4.

Chapter 2

Channel Model

In this chapter, we describe the two channel models we will use in this thesis. The models are simple descriptions of a single-carrier passband communications system: they take phase offsets into account between the received carrier and the local carrier in the receiver, but disregard timing offsets.

We organized this chapter as follows. First, we review some basic notions from digital communications, with special emphasis on single-carrier passband communications systems. We present our first model, in which the phase offset is constant. We then investigate how noise sources such as thermal and shot noise amount to random fluctuations in oscillator and clock signals. At the end of this chapter, we formulate the second signal model, which incorporates random phase fluctuations, i.e., the phase drift is modeled as a Gaussian random walk. We discuss under which conditions both models are valid.

2.1 Digital Communications System

In this section, we review some basic concepts from digital communications. For a classical treatment of this subject, see e.g. [165]. We will follow the exposition in [203].

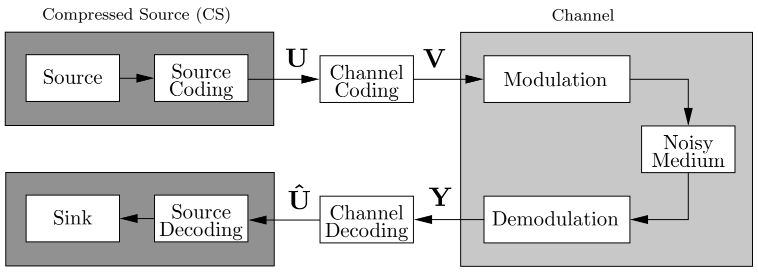


Figure 2.1: Digital data transmission over a noisy channel.

Assume that we are interested in transmitting data (such as images, audio or video signals) from a point A to a point B or that we wish to store some information that we would like to retrieve later on. In both cases we desire that the received and the retrieved data is identical to the data transmitted or stored, or, if errors cannot be avoided, that there are as few errors as possible.

Fig. 2.1 shows the typical blocks of a model for digital data transmission over a noisy channel:

- **(Source/source coding)**
  The data we would like to transmit is produced by a source. Its output can be compressed (losslessly or lossy) by a source coding scheme to reduce the amount of data to be transmitted. The output of such a compressed source may for example be modeled as a binary i.i.d. source. By $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, where $u_i \in \mathcal{U}$, we denote the vector of k consecutive source output symbols.

- **(Channel coding)**
  In order to increase the reliability of the transmission of the signal through the (noisy) channel, the channel encoder introduces redundancy to the data coming out of the compressed source.

  A trivial form of encoding is to repeat each source symbol m times, where m is a positive integer. In non-trivial encoding schemes, the source output vector $\mathbf{u}$ of length k is mapped to a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ of length n where $v_i \in \mathcal{V}$. The art of channel coding is to find a "good" subset $\mathcal{C}$ of $\mathcal{V}^n$ called codebook or, in short, code; the elements of $\mathcal{C}$, referred to as the codewords, should be as far "apart" from each other as possible.

For more information on coding, we refer the reader to Appendix C.

- **(Modulation/noisy medium/demodulation)**
  The noisy medium is the physical medium that is used to send the data from the transmitter to the receiver. In wireless transmission, the data is transmitted in free space;[1] other media are wire lines, optical fiber cables, and magnetic materials (as, e.g., in hard drives). The medium corrupts the signal by a variety of mechanisms as for example (1) thermal noise and shot noise generated by electronic devices; (2) interference with background signals, such as signals stemming from other users or power-line signals.

  Before the data $\mathbf{v}$ can be sent over the noisy medium, it must be converted into analog waveforms that match the characteristics of the medium; this is the task of the modulator. Typically, the code symbols $\mathbf{v}$ are first converted into channel symbols $\mathbf{x}$. This conversion is generally referred to as line encoding. The channel symbols $\mathbf{x}$ are modulated on analog waveforms. At the receiving end of the communications system, the demodulator transforms the received (analog) signal into a sequence $\mathbf{y}$ of symbols.

  The concatenation of the three blocks modulation, noisy medium, and demodulation is called the communications channel (or channel for short). A communications channel is characterized by the channel law $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ which is the conditional probability of $\mathbf{y}$ given $\mathbf{x}$.

  When the output y_k of the channel solely depends on the *current* input x_k and is conditionally independent of *previous* inputs and outputs, then the channel is called "memoryless". The channel law $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ can then be written as

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \stackrel{\triangle}{=} \prod_{k=1}^{n} P_{Y|X}(y_k|x_k), \tag{2.1}$$

  under the assumption that the channel is used without feedback. The channel is thus fully characterized by the conditional probability function $P_{Y|X}$ and can be represented by a diagram as in

[1]We remind the reader of the fact that free space is not a medium in the *physical* sense. We use the word "medium" here in a more abstract sense, as is standard in the communications literature.

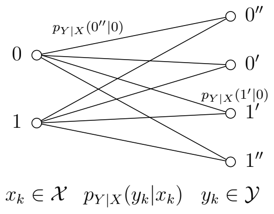


Figure 2.2: Example of a channel diagram.

Fig. 2.2. An example of memoryless channel model is the additive white Gaussian noise channel with binary (BI-AWGNC) or continuous (C-AWGNC) inputs. The input to the BI-AWGNC is ± 1, the output is a real number, and the quality of the channel is characterized by the variance of the additive white Gaussian noise; in the C-AWGN channel, the input is a real number.

The efficiency of the information transmission over the channel is quantified by the transmission rate R (in bits per channel use) defined as the ratio

$$R \triangleq \frac{\log_2 |\mathcal{C}|}{n} = \frac{k \log_2 |\mathcal{U}|}{n}. \tag{2.2}$$

We provide more information about the channel in Section 2.2.

- **(Channel decoding)**
  The sequence $\mathbf{y}$ is passed to the channel decoder, which tries to eliminate errors that might have occurred during transmission; thereby, it exploits the redundancy contained in the received data. Based on the observation $\mathbf{y}$ our estimate about $\mathbf{u}$ is $\hat{\mathbf{u}} = (\hat{u}_1, \ldots, \hat{u}_k)$ of length k with $\hat{u}_i \in \mathcal{U}$. Instead of estimating $\mathbf{u}$, we are possibly interested in an estimate $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n)$ about $\mathbf{v}$, where $\hat{v}_k \in \mathcal{V}$. Channel decoding is the art of finding a "good" and efficient decoding algorithm for a given channel code and a given channel. By "good" we mean that the probability of error $\Pr[\mathbf{u} \neq \hat{\mathbf{u}}]$ should be as small as possible.

- **(Source decoding/sink)**
  The source decoder uncompresses $\hat{\mathbf{u}}$ and delivers the result to the final destination ("sink").

2.2 The Communications Channel

We defined the channel as the concatenation of modulation, the noisy medium, and demodulation (see Fig. 2.1); the modulator transforms the digital information into analog waveforms, which are transmitted over the noisy medium, and converted back into a sequence of symbols by the demodulator at the receiver side.

Two major classes of analog waveforms are **baseband** and **passband** signals, each leading to substantially different transmitter and receiver structures. Baseband signals are pulse trains, the information is encoded in the amplitude of the pulses. They are used in applications such as Integrated Services Digital Networks (ISDN), Local Area Networks (LANs), and digital magnetic recording systems. In passband communications systems, a baseband signal is modulated unto a sinusoidal carrier, such that the resulting waveform fits into the frequency range available for transmission. In radio, wireless and satellite communications systems, information is transmitted by means of passband signals.

In the following, we briefly outline both classes of communications systems; we refer to [165] for more detailed information. We will assume that the channel code $\mathcal{C}$ is binary. We will use the symbol $\mathbf{b}$ (instead of $\mathbf{v}$) for the encoded bits. We will also assume that the noisy medium solely adds white noise $n(t)$ to the transmitted signal $s(t)$. The received signal is then given by

$$r(t) = s(t - \tau_C) + n(t), \tag{2.3}$$

where τ_C is the delay of the channel.

2.2.1 Baseband System

A rudimentary block diagram of a baseband communications system is depicted in Fig. 2.4.

Modulation

The **line encoder** maps sequences of $\log_2 M$ encoded bits b_k to channel symbols x_k taking value in $\{\pm 1, \pm 3, \ldots, \pm (M-1)\}$. The sequence $\mathbf{x}$ of channel symbols passes through a linear filter (**transmit filter**) with impulse response $g_T(t)$. The resulting (base-band) waveform (see Fig. 2.3) has the form

$$s_{\text{BB}}(t) = \sum_k x_k \, g_T(t - kT), \tag{2.4}$$

where T is the inverse of the symbol rate. The signal $s_{\text{BB}}(t)$ is transmitted over the noisy medium.

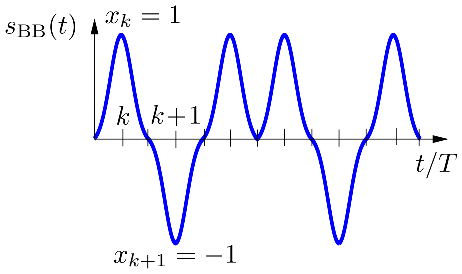


Figure 2.3: Baseband signal.

Demodulation

The received signal is processed by a linear filter (**receiver filter** or "matched filter") whose impulse response g_R is given by $g_R(t) \triangleq g_T(\tau_R - t)$, where τ_R is a chosen such that $g_T(\tau_R - t)$ is a causal function.

In addition, the convolution $h(t) \triangleq [g_T \star g_R](t - \tau_R)$ often satisfies the first Nyquist criterion

$$h(kT) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for } k \neq 0. \end{cases} \tag{2.5}$$

An extensively used class of functions that satisfy (2.5) are the raised cosine pulses

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T} \frac{\cos(\alpha \pi t/T)}{1 - 4\alpha^2 t^2/T^2}, \tag{2.6}$$

where the parameter α is called the roll-off factor and satisfies $0 < \alpha \leq 1$.

The output of the matched filter is sampled at a rate $1/T$; the **line decoder** converts the samples in the sequence $\mathbf{y}$, which will be further processed by the channel decoder.

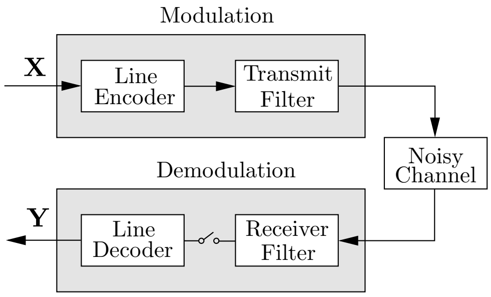


Figure 2.4: Baseband communications channel.

We investigate the signals in the baseband receiver in more detail. The received waveform $r(t)$ is given by

$$y(t) = \sum_{\ell} x_\ell \, g_T(t - \ell T - \tau_C) + n(t), \tag{2.7}$$

and the output of the receiver filter equals

$$\tilde{y}(t) = \sum_{\ell} x_\ell \, h(t - \ell T - \tau_C - \tau_R) + \tilde{n}(t), \tag{2.8}$$

where $\tilde{n}(t) \triangleq [n \star g_R](t)$. The signal $\tilde{y}(t)$ is sampled at the instances $t = \hat{\tau} + kT$ resulting in the samples

$$y_k = \sum_{\ell} x_\ell \, h(\tau + (k - \ell)T) + n_k, \tag{2.9}$$

where $n_k = \tilde{n}(\hat{\tau} + kT)$, $y_k \triangleq \tilde{y}(kT)$, and $\tau \triangleq \hat{\tau} - \tau_R - \tau_C$ is the timing offset. If

$$\hat{\tau} = \tau_{\text{ideal}} \triangleq \tau_R + \tau_C, \tag{2.10}$$

the timing offset τ equals 0, and as a consequence of the first Nyquist criterion (2.5), the expression (2.9) simplifies to

$$y_k = x_k + n_k. \tag{2.11}$$

The sample y_k then only depends on the channel symbol x_k; otherwise, it is (in principle) affected by *all* channel symbol $\mathbf{c}$ (cf. (2.9)), an effect called "inter-symbol interference" (ISI). In many practical receivers, a timing-synchronization algorithm (see Fig. 2.5) tries to align the sampler to the incoming signal in order to circumvent ISI. However, (small) deviations between $\hat{\tau}$ and τ_{ideal} are unavoidable. An alternative approach is not to adjust $\hat{\tau}$ at all, but to use a free-running sample clock instead. The timing offset τ is then estimated from the samples $\mathbf{r}$, and the ISI is compensated for by digital signal processing.

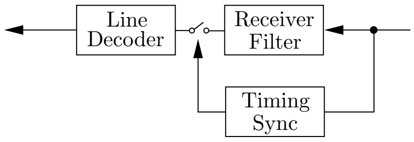


Figure 2.5: Timing synchronization in a baseband receiver.

2.2.2 Passband System

Fig. 2.6 shows a basic block diagram of a (single-carrier) passband communications system.

Modulation
Channel symbols $\mathbf{x}$ are passed through a linear filter $g_T(t)$; the resulting baseband signal $s_{\text{BB}}(t)$ is modulated onto a sinusoid $c(t)$ with frequency f_C

$$c(t) \triangleq e^{j2\pi f_C t}, \tag{2.12}$$

amounting to the passband signal $s_{\text{PB}}(t)$

$$s_{\text{PB}}(t) = \text{Re}\{s_{\text{BB}}(t) e^{j2\pi f_C t}\}. \tag{2.13}$$

Before the signal $s_{\text{PB}}(t)$ is transmitted over the communications channel, it is fed into a bandpass filter.

Depending on the structure of the baseband signal $s_{\text{BB}}(t)$, one distinguishes two different classes of passband modulation schemes: non-offset modulation and offset modulation. In **non-offset modulation**, the baseband signal $s_{\text{BB}}(t)$ has the form

$$s_{\text{BB}}(t) = \sum_k x_k \, g_T(t - kT), \tag{2.14}$$

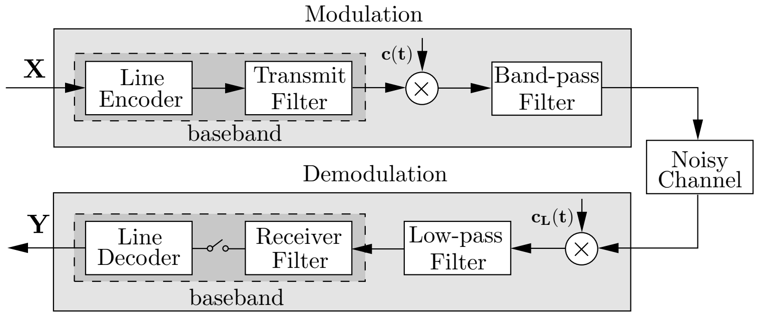


Figure 2.6: Passband communications channel.

where $g_T(t)$ is the signaling pulse. Quadrature amplitude modulation (QAM) and phase shift keying (PSK) are the two most wide-spread non-offset modulation schemes. With QAM modulation, x_k in (2.14) has the form

$$x_k = a_k + jb_k \tag{2.15}$$

with a_k and b_k belonging to $\{\pm1, \pm3, \ldots, \pm(M-1)\}$. With PSK, we have

$$x_k = e^{j\alpha_k}, \tag{2.16}$$

with $\alpha_k \in \{0, 2\pi/M, \ldots, 2\pi(M-1)/M\}$. In **offset modulation**, the baseband signal s_{BB} is given by

$$s_{\mathrm{BB}}(t) = \sum_k a_k g_T(t-kT) + j\sum_k b_k g_T(t-kT-T/2). \tag{2.17}$$

In offset quadriphase modulation (OQPSK), which is the most common offset modulation scheme, a_k and b_k in (2.17) take values ±1 (as in 4-QAM).

The encoded bits $\mathbf{b}$ can be mapped to (channel) symbols $\mathbf{x}$ in a number of ways. The most common mapping is **Gray encoding**, where adjacent (M-ary) signal amplitudes differ by one binary digit, as illustrated in Fig. 2.7. The most likely errors caused by noise involve the erroneous selection of an adjacent amplitude to the transmitted amplitude x_k. In such a case, only a single-bit error occurs in the M-bit sequence.

Demodulation

The received signal is first processed by a pre-filter (not shown in Fig. 2.6) to eliminate out-of-band noise. Its output is down-converted to a baseband signal: first, it is multiplied by a local carrier $c_L(t)$, then it is passed through a low-pass filter. The output of this filter is processed by a baseband-receiver.

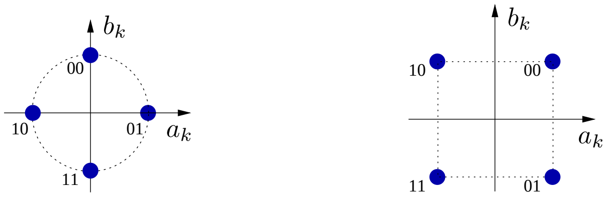


Figure 2.7: Non-offset modulation.
(left) 4-PSK; (right) 4-QAM, a.k.a Q(uadrature) PSK.

We now have a closer look at the signal processing in the passband receiver. We consider non-offset modulation, the extension to offset modulation is straightforward. We merely focus on carrier synchronization and put aside timing synchronization, since it has been considered previously.

The output of the low-pass filter in Fig. 2.4 can be represented as the complex signal

$$y(t) = e^{j(2\pi\nu t+\theta)} s_{\mathrm{BB}}(t) + \tilde{n}(t), \tag{2.18}$$

where

- $\tilde{n}(t)$ is low-pass noise, whose bandwidth is usually much wider than the bandwidth of the baseband signal $s_{\mathrm{BB}}(t)$.
- $\nu \triangleq f_R - f_L$ is referred to as the carrier frequency offset,
- f_L is the frequency of the local reference and f_R is the frequency of the *incoming* carrier; due to the Doppler effect and clock instabilities[2], the frequency f_R may differ from f_C, the frequency of the *transmitted* carrier,
- $\theta \triangleq \theta_L - \theta_R - 2\pi f_R \tau$ is a phase offset,

[2]We address clock instabilities in Section 2.4.

- θ_L and θ_R is the phase of the local reference and the incoming carrier respectively,
- we assumed that the low-pass filter has a unity frequency response for the low-pass signal components.

The output of the receiver filter $\tilde{y}(t) = [g_R \star y](t)$ is given by

$$\begin{aligned}\tilde{y}(t) \;\; &\triangleq \;\; \sum_\ell x_\ell \int_0^\infty e^{j(2\pi\nu(t-t')+\theta)} g_T(t - t' - \ell T - \tau_C) \\ &\qquad\qquad g_T(\tau_R - t')dt' + \tilde{m}(t), \end{aligned} \tag{2.19}$$

where $\tilde{m}(t) \triangleq [g_R \star \tilde{n}](t)$. The expression (2.19) can be approximated as

$$\tilde{y}(t) \;\; \approx \;\; e^{j(2\pi\nu t+\theta)} \sum_\ell x_\ell \, h(t - \ell T - \tau_C - \tau_R) + \tilde{m}(t), \tag{2.20}$$

as long as the phase $\theta(t) \triangleq 2\pi\nu t + \theta$ varies only little over a time interval of length T. Sampling $\tilde{y}(t)$ at the "ideal" instances $t = \tau_{\text{ideal}} + kT \triangleq \tau_R + \tau_C + kT$ yields

$$y_k = e^{j(2\pi\nu(\tau_{\text{ideal}}+kT)+\theta)} x_k + n_k, \tag{2.21}$$

where $n_k = \tilde{n}(\tau_{\text{ideal}} + kT)$. Note that carrier offsets (in contrast to timing offsets) do not lead to inter-symbol interference, as long as the phase offset remains small. Most passband receivers are equipped with algorithms to track the carrier offsets ν and τ, as depicted in Fig. 2.8.[3] In addition, they often contain algorithms to correct for timing offsets.

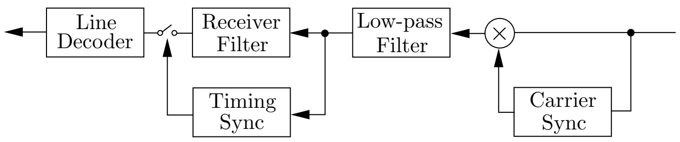


Figure 2.8: Carrier and timing synchronization in a passband receiver.

[3]The picture shows one possible architecture; there is a large variety of classical synchronization schemes, see [135] [136].

2.3 Constant-Phase Model

We propose a first simple stochastic model for a single-carrier communications systems.

Constant-phase model:

$$Y_k = X_k e^{j\Theta} + N_k \tag{2.22}$$

with $\Theta \in [0, 2\pi)$ and

$$N_k \sim \mathcal{N}_{0,\sigma_N^2}. \tag{2.23}$$

The above model is a *stochastic* model, its variables (e.g., Θ) are *random* variables. We use capital letters to denote random variables and small letters for their realizations.

The model (2.22) (2.23) is valid if

a) a timing-synchronizer tracks the timing offsets, and hence the expression (2.21) is a good description of the received symbols y_k,

b) the frequency offset $\nu = 0$,

c) the phase offset $\theta \stackrel{\triangle}{=} \theta_L - \theta_R - 2\pi f_R \tau$ is constant.

The last assumption is typically not met. The phase offset often undergoes random fluctuations. In the following section, we study how noise sources such as thermal, shot and flicker noise amount to phase instabilities. In Section 2.5, we present a simple model that takes those instabilities into account.

2.4 Phase Noise

Clock and oscillator signals that occur in communications systems are not perfectly periodic. Practical clocks, such as CMOS LC-oscillators for example (see e.g., [98] [64] [215]), are affected by phase and frequency instabilities called *phase noise*. This is even the case for high-precision frequency sources such as quartz oscillators, masers and passive atomic frequency standards [135, p. 142].

Since we are interested in modeling the impact of phase offsets in passband systems, it is of crucial importance to have a good understanding of the random fluctuations in the phase of an oscillator. In this section, we present several models for phase noise in free-running and forced oscillators.

The oscillator output is typically perturbed by short-term and long-term instabilities. Long-term instabilities, also known as drifts or trends, may be due to aging of the resonator material (e.g., in quartz oscillators). These usually very slow changes are much less critical than the short-term instabilities, caused by noise sources such as thermal, shot, and flicker noise in electronic components. The oscillator output may also interfere with other signals. For example, in highly integrated oscillator circuits, switching signals from the digital portion of the circuit can couple with the clock signal through the substrate or power supply lines (see e.g., [81]). This kind of interference can often be avoided by careful system design.

Phase noise degrades the performance of communications systems, as we illustrate by two examples. The transitions in an oscillator signal are sometimes used as a time reference (e.g., the sampler clock in a baseband receiver). The spacing between those transitions is ideally constant; in practice, however, they will be variable due to phase noise (see Fig. 2.12). The randomness in the transition times, called *timing jitter*, has a harmful effect on the sampling process, as illustrated in Fig. 2.9: the uncertainty in the sampling times translates directly to uncertainty in the sampled value.

Phase noise also has a deteriorating influence on the down-conversion in passband receivers (see Fig. 2.10). The power spectral density of a periodic signal (with period T_s) consists of Dirac deltas located at the harmonics k/T_s ($k = 1, 2, \dots$). Due to phase noise, power “leaks” from the harmonics to neighboring frequencies: the Dirac deltas becomes smooth “bumps”, centered at the harmonics. As a consequence, background signals in the frequency band adjacent to the incoming data signal are down-converted and interfere with the desired baseband signal (2.18). This phenomenon is called *interchannel interference.*

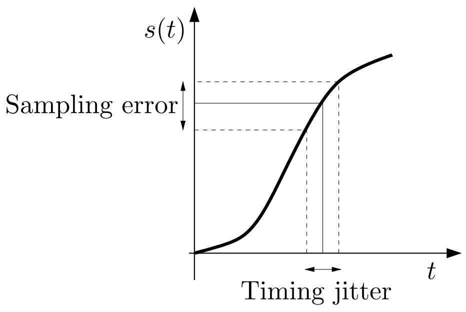


Figure 2.9: Sampling error due to timing jitter.

Oscillators are omnipresent in communications and optical systems and as a consequence, phase noise has been studied intensively in the past. Nevertheless, a generally accepted model for phase noise does not seem to exist. The same holds for flicker noise, despite of its ubiquity. The modeling of phase noise, in fact, of noise in general, remains a very active research field.

In the following, we briefly review the three most common noise sources: thermal, shot and flicker noise. We derive a simple model for phase noise in free-running oscillators, i.e., oscillators that are *not* locked unto a reference. Our aim is to gain some insight, not to derive a complete theory. We review some modeling approaches that have been proposed in the literature, for free-running clocks and for phase-locked loops. At the end of this section, we discuss simple heuristic models for phase noise.

Common noise sources

- **(Shot noise)**
  Shot noise consists of random fluctuations of electric currents in resistors, pn-junctions, transistors, and other electronic devices. The fluctuations are due to the fact that an electric current is carried by discrete charges (electrons and holes). The power spectral density of this (additive) noise source is constant for a very large frequency range (tens of hertz to gigahertz). It can be represented as a zero-

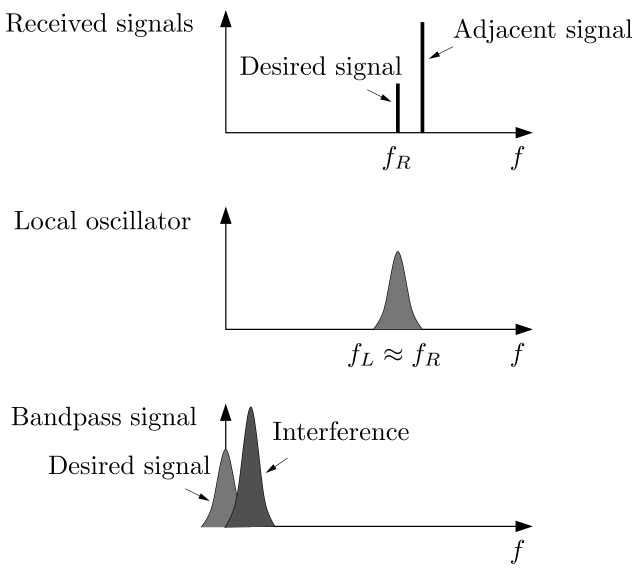


Figure 2.10: Interchannel interference.

mean white noise source with spectral density

$$S_{\text{shot}}(f) = 2qI, \tag{2.24}$$

where q is the electron charge, I the (average) current and f the frequency. The expression (2.24) can be derived as follows.[4] The flow of electrical charges through some spatial section can be modeled as a random pulse train. The corresponding (stochastic) current $I(t)$ is of the form:

$$I(t) = \sum_k q\delta(t - t_k), \tag{2.25}$$

where t_k is the arrival time of the k-th carrier, and q is the charge of each carrier. We model the arrival of the carriers as a Poisson process with rate λ. The signal $I(t)$ is an i.i.d. stochastic process with mean

$$I \triangleq \mathrm{E}\big[I(t)\big] = \lambda q. \tag{2.26}$$

[4]We follow in part the exposition in [175]; a more detailed microscopic model can be found in [218, pp. 54–68].

The shot noise, i.e., the random fluctuations around the average current $I(t)$, is given by the zero-mean current $\tilde{I}(t)$:

$$\tilde{I}(t) \triangleq I(t) - I. \tag{2.27}$$

The power spectral density of $\tilde{I}(t)$ follows from Carson's theorem [218, pp. 22–23].

Theorem 2.1. (Carson's theorem)
Let $x(t)$ be defined as follows:

$$X(t) = \sum_k A_k\, g(t - t_k), \tag{2.28}$$

where A_k are i.i.d. random variables, the function $g(t)$ ("pulse") has the Fourier transform $G(f)$, and the number of pulses in a given time interval is Poisson distributed with rate λ. The power spectral density of $X(t)$ is given by

$$S_x(f) = 2\lambda \mathrm{E}\big[A^2\big]\,|G(f)|^2 + 4\pi \mathrm{E}^2[x]\delta(f). \tag{2.29}$$

We apply Carson's theorem to the signal $\tilde{I}$ (2.27) with $A_k = q$ and $G(f) = 1$, and obtain:

$$\begin{aligned} S_x(f) &= 2\lambda q^2 && (2.30)\\ &= 2qI, && (2.31) \end{aligned}$$

which is the spectral density (2.24).

- **(Thermal noise)**
A conductor in thermal equilibrium with its surroundings exhibits random fluctuations even when no (average) current is flowing through it. These fluctuations were first observed by Johnson in 1928 [92], and shortly afterwards, Nyquist proposed a theoretical explanation [154]. This noise source is often called Johnson-Nyquist noise (or "thermal noise"). The power spectral density of the open-circuit voltage and closed-circuit current is given by

$$S_{v,\text{thermal}}(\omega) = 4k_B TR \quad \text{and} \quad S_{i,\text{thermal}}(\omega) = 4k_B T/R \tag{2.32}$$

respectively, where k_B is the Boltzmann constant, T is the equilibrium temperature and R is the resistance. In most standard accounts on noise, thermal noise and shot noise are treated as fundamentally different physical processes. Sarpeshkar and al. [175] have shown that both noise sources are in fact the result of one and the same physical mechanism, i.e., the discrete nature of charge transfer; we shortly review the line of thought in [175].

If no voltage is applied across a resistor, electrons flow in the resistor due to diffusion. The resulting forward and backward current $I_f(t)$ and $I_b(t)$ can be modeled as Poisson processes. Both currents cancel on the average, since the total average current is assumed to be zero. If the currents $I_f(t)$ and $I_b(t)$ are statistically independent, they generate shot noise with total power spectral density $S_{\text{shot}}(f) = 2q(I_f + I_b)$ (cf. (2.24)), where I_f and I_b are the average forward and backward current respectively. The latter are proportional to the concentration of the carriers n and to the area of the cross section, and inversely proportional to the length L of the resistor:

$$I_f = I_b = qDnA/L, \tag{2.33}$$

where D is the so-called diffusion constant. By means of (2.33) and Einstein's relation $D/\mu = kT/q$, where μ is the mobility constant, the noise power can be written as:

$$\begin{aligned} S_{\text{shot}}(f) &= 2q(I_f + I_b) && (2.34)\\ &= 4q^2 DnA/L && (2.35)\\ &= 4(q\mu n)kTA/L && (2.36)\\ &= 4kT\sigma A/L && (2.37)\\ &= 4kT/R && (2.38)\\ &= S_{i,\text{thermal}}(\omega), && (2.39) \end{aligned}$$

where σ is the conductivity of the material. The key step in the derivation is the use of Einstein's relation (cf. (2.36)). We obtained thus Johnson's and Nyquist's result (2.32) for the power spectral density of short-circuit noise in a resistor. In other words, thermal noise is nothing but shot noise, originating from the discreteness of the forward and backward current.

- **(Flicker noise)**
  In a large variety of unrelated systems, signals have a power spectral density that is proportional to $1/f$ at low frequencies f. Such signals are called "$1/f$ noise" (or "flicker noise" or "pink noise").

Examples include music and speech, the current through ion channels, network and freeway traffic, currents and voltages in electronic devices, etc. (see [95] and references therein). Whereas thermal and shot noise are fairly well understood, there is still quite some controversy around $1/f$ noise. Recently, Kaulakys et al. [95] [96] proposed a simple and generic mathematical description for this type of noise—in our opinion, the most plausible and promising of all approaches we are aware of. Kaulakys et al. argue that the origin of $1/f$ noise lies in Brownian fluctuations of the *interevent time* of signal pulses. In the following, we briefly review Kaulakys' model [95] [96].

We consider again a signal $x(t)$ of the form (2.28). The pulse *shape* $g(\cdot)$ mainly influences the high-frequency portion of the power spectral density, and is not responsible for $1/f$ noise. Fluctuations in the pulse amplitude a_k usually result in white or Lorentzian noise, but not $1/f$ noise. For the purpose of this analysis, we restrict ourselves to the noise due to correlations between the transit times t_k: we choose a Dirac delta as pulse shape and assume that the pulse amplitude is constant, i.e., $a_k = a$, for all k, resulting in the signal $x(t)$:

$$x(t) = \sum_k a\delta(t - t_k). \tag{2.40}$$

This model corresponds to the flow of identical *point* objects such as electrons, photons, etc. Suppose now that the recurrence times $\tau_k \triangleq t_k - t_{k-1}$ follow a first-order autoregressive model:

$$\tau_k = \tau_{k-1} - \gamma(\tau_{k-1} - \bar{\tau}) + \sigma n_k, \tag{2.41}$$

where σ ("noise variance") and γ ("damping factor") are "small" positive real numbers, n_k is an i.i.d. Gaussian random variable with zero mean and unit variance, and $\bar{\tau}$ is the steady state value of τ_k. Note that for large k the variance of the interevent time τ_k converges to a finite value:

$$\text{Var}[\tau_k] = \sigma^2/(2\gamma). \tag{2.42}$$

In [95] it is shown that the power spectral density of the signal $x(t)$ (2.40) with interevent times (2.41) for $t \gg \gamma^{-1}$ and $f_2 \triangleq \sqrt{\gamma}/(\pi\sigma) > f > f_1 \triangleq \gamma^{3/2}/(\pi\sigma)$ has the form

$$S(f) = I^2 \frac{\alpha}{f}, \tag{2.43}$$

where $I = a/\bar{\tau}$ and α is a dimensionless constant:

$$\alpha = \frac{2}{\sqrt{\pi}} K e^{-K^2}, \tag{2.44}$$

with $K = \bar{\tau}\sqrt{\gamma}/\sigma$. The model amounts to a $1/f$ spectrum in a wide frequency range (f_1, f_2) with $f_2/f_1 = \gamma^{-1}$. As a result of the non-zero damping factor γ and, consequently, due to the finite variance of the recurrence time τ_k, the model is free from the unphysical divergence at $f = 0$. For $f < f_1$ the power spectrum density is Lorentzian [95]:

$$S(f) = I^2 \frac{4\tau_{\mathrm{rel}}}{1 + \tau_{\mathrm{rel}}^2 \omega^2}, \tag{2.45}$$

where $\omega = 2\pi f$ and $\tau_{\mathrm{rel}} = \sigma^2/(2\bar{\tau}\gamma^2)$. For $0 \leq f < 1/(2\pi\tau_{\mathrm{rel}})$, the power spectral density is flat, i.e., $S(f) = 4I^2\tau_{\mathrm{rel}}$ and $S(0)$ is finite.

Recently, Kaulakys [96] proposed a non-linear stochastic differential equation for the signal (2.40) with interevent times (2.41), and $\gamma = 0$:

$$\frac{dx}{dt} = \frac{\sigma^2}{a^3} x^4 + \frac{\sigma}{a^{3/2}} x^{5/2} n(t), \tag{2.46}$$

where $n(t)$ is zero mean white Gaussian noise with unit variance. We refer to [96] for more details. A similar differential equation for $\gamma > 0$ has not been derived yet.

We now investigate how noise sources such as thermal, shot and flicker noise can generate phase noise. First we consider free-running clocks, then phase-locked loops.

2.4.1 Phase Noise in Free-Running Clocks

A Simple Model

We consider a general autonomous (continuous-time) system described by the first-order differential equation

$$\frac{dx}{dt} = f(x), \tag{2.47}$$

where $x \in \mathbb{R}^n$ is the state of the system and $f : \mathbb{R}^n \to \mathbb{R}^n$. We assume that f satisfies the conditions of the existence and uniqueness theorem for initial value problems, and that the system (2.47) has a non-trivial T_s-periodic solution $x_s(t)$. The path P_s (or "orbit" or "limit cycle") is defined as

$$P_s = \{x \in \mathbb{R}^n : x = x_s(t), t \in [0, T_s)\}. \tag{2.48}$$

The distance $d(x, P_s)$ between a point $x \in \mathbb{R}^n$ and the path P_s is defined as

$$d(x, P_s) \overset{\triangle}{=} \min_{x' \in P_s} |x - x'|. \tag{2.49}$$

The solution $x_s(t)$ is assumed to be an *attractor*, which means that there exists a number $\varepsilon > 0$ such that with each initial value x_0 satisfying $d(x_0, P_s) < \varepsilon$, there corresponds an asymptotic phase $\theta(x_0) \in \mathbb{R}$ with the property

$$\lim_{t \to \infty} |x(t, x_0) - x_s(t + \theta(x_0))| = 0, \tag{2.50}$$

where $x(t, x_0)$ is the (unique) solution of (2.47) with attains the value x_0 at $t = 0$. In other words, as soon all noise sources are "switched off", the state $x(t)$ will return to the orbit P_s; deviations from the orbit ("amplitude deviations") are smoothed out by a control mechanism, but a phase offset θ persists, even when no further perturbations occur!

The fact that the system (2.47) is supposed to have a stable periodic solution immediately implies that f is a *non-linear* function. Indeed, linear systems do not have *stable* (or "attracting") limit cycles. This inherent non-linearity makes the study of phase noise notoriously difficult.

We are interested in the response of the system (2.47) to a "small" additive stochastic perturbation $N(t)$; the perturbed system is given by

$$\frac{dx}{dt} = f(x) + N(t). \tag{2.51}$$

As a first step in our analysis, let us consider the response $x(t)$ to a small deterministic perturbation $n_0 \in \mathbb{R}^n$ at $t = 0$, as illustrated in Fig. 2.11. We assume that the state $x(t)$ evolves along the path P_s before the perturbation occurs, more precisely, $x(t) \overset{\triangle}{=} x_s(t)$ for all $t \in [-t_0, 0)$ with $t_0 > 0$. The response $x(0)$ at $t = 0^+$ equals:

$$x(0) = x_s(0) + n_0 \overset{\triangle}{=} x_0. \tag{2.52}$$

Since the limit cycle P_s is an attractor, the state $x(t)$ will return to P_s:

$$x(t) = x_s(t + \theta(x_s(0), n_0)), \qquad (t \gg 0) \tag{2.53}$$

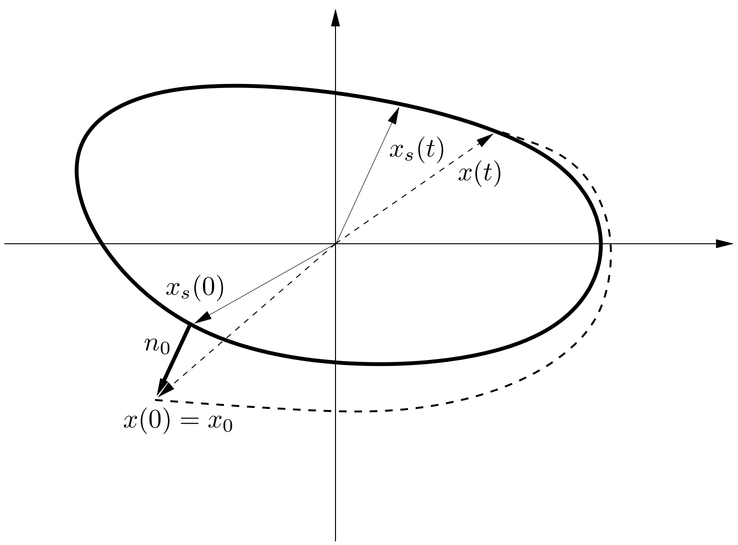


Figure 2.11: Perturbation of the stable orbit $x_s(t)$.

where the asymptotic phase shift $\theta(x_s(0), n_0)$ depends on the initial position $x_s(0)$ and the perturbation n_0. The form of the function $\theta(\cdot)$ depends on the function $f(\cdot)$ (cf. (2.47)). We consider now the Taylor-expansion of $\theta(x_s(0), n_0)$ around $n_0 = 0$:

$$\theta(x_s(0), n_0) = \theta(x_s(0), 0) + \nabla_{n_0}\theta(x_s(0), n_0)\big|_{n_0=0} \cdot n_0 + \mathcal{O}(|n_0|^2) \tag{2.54}$$

$$= \nabla_{n_0}\theta(x_s(0), n_0)\big|_{n_0=0} \cdot n_0 + \mathcal{O}(|n_0|^2), \tag{2.55}$$

where we have used the fact that $\theta(x_s(0), 0) = 0$. Since the perturbation n_0 is assumed to be small, one can (to a good approximation[5]) omit the non-linear terms in (2.54):

$$\theta(x_s(0), n_0) \approx \gamma(x_s(0)) \cdot n_0, \tag{2.56}$$

[5]Hajimiri et al. have performed SPICE-simulations to verify the linearity assumption. They concluded that the assumption holds as long as the perturbation is smaller than 10% of the signal amplitude $x_{\max} = \max_t |x_s(t)|$ [78, p. 38]. We also investigated the linearity assumption by means of simulations (i.e., by numerical integration of (2.51)), and obtained similar results. The effective injected charges due to actual noise and interference sources in practical circuits are typically of the order 10^{-4} of $x_{\max}$ or even smaller, hence the linearity assumption is usually valid.

where

$$\gamma(x_s(0)) \triangleq \nabla_{n_0}\theta(x_s(0), n_0)|_{n_0=0}\,. \tag{2.57}$$

Suppose now that the system is disturbed by multiple deterministic perturbations $n_1, n_2, \ldots, n_M$ at the time instances $t_1, t_2, \ldots, t_M$ respectively. For simplicity, we assume that the time instances t_i are "far" apart such that the state x has reached the orbit P_s before each perturbation n_i occurs.[6] The resulting asymptotic phase shift θ for $t \gg t_M$ depends on the states $x_s(t_i^- + \theta(t_i^-))$ at t_i^- and the perturbations n_i $(i = 1, \ldots, M)$. Again, we assume that the perturbations are small and that θ depends linearly on the disturbances n_i

$$\theta = \sum_{i=1}^{M} \gamma\big(x_s(t_i^- + \theta(t_i^-))\big) \cdot n_i. \tag{2.58}$$

Since the time t uniquely determines the position of $x_s(t)$ along the path P_s, we will from now on write $\gamma(t_i^- + \theta(t_i^-))$ instead of $\gamma\big(x_s(t_i^- + \theta(t_i^-))\big)$.

The same reasoning can be applied to the model (2.51), which results in the following continuous-time phase model.

Continuous-time phase noise model:

$$\Theta(t) = \int_{-\infty}^{t} \gamma(t' + \Theta(t')) \cdot N(t')dt'. \tag{2.59}$$

In words:

(Key property of phase noise)
Phase noise is intrinsically an **accumulative** process; it results from **integrating** the system's perturbations over time.

The phase $\Theta(t)$ (cf. (2.59)) is a random variable. Note that (2.59) is a *nonlinear* stochastic differential equation.

[6] If this assumption does not hold, our theory is still valid, but the reasoning becomes a bit more involved.

Suppose now that the noise source $N(t)$ is an *i.i.d. process* with (finite) variance σ_0^2, as for example thermal and shot noise.[7] We assume that t is sufficiently large so that the initial condition has been "forgotten" and the pdf $p_\Theta(\theta(t))$ is uniform. The random variable $K(t) \triangleq \gamma(t+\Theta(t))N(t)$ has zero mean, and as a consequence of (2.59), the same holds for the phase $\Theta(t)$. The variance $\mathrm{Var}[\Theta(t)]$ of the phase grows linearly with time, which can be shown as follows. Since $N(t)$ is i.i.d.,

$$\mathrm{E}\big[(\Theta(t+\Delta t)-\Theta(t))^2\big] = \sigma_0^2 \int_t^{t+\Delta t} \mathrm{E}\Big[|\gamma(t'+\Theta(t'))|^2\Big] dt', \qquad (2.60)$$

where Δt is a "small" positive real number. The expectation in the RHS of (2.60) does not depend on t (for large t):

$$\mathrm{E}\Big[|\gamma(t+\Theta(t))|^2\Big] \triangleq e_0. \qquad (2.61)$$

As a consequence,

$$\mathrm{E}\big[(\Theta(t+\Delta t)-\Theta(t))^2\big] = \sigma_0^2\, e_0\, \Delta t, \qquad (2.62)$$

and, therefore, the derivative of $\mathrm{Var}[\Theta(t)]$ is constant:

$$\frac{d\mathrm{Var}[\Theta(t)]}{dt} \triangleq \lim_{\Delta t \to 0} \mathrm{E}\big[(\Theta(t+\Delta t)-\Theta(t))^2/\Delta t\big] = \sigma_0^2\, e_0. \qquad (2.63)$$

Assuming $\mathrm{Var}[\Theta(0)] = 0$, we obtain

$$\mathrm{Var}[\Theta(t)] = c_0 t, \qquad (2.64)$$

where $c_0 \triangleq \sigma_0^2\, e_0$. As a consequence of (2.64), spectrograms[8] of the phase Θ (measured over a large, but finite time interval) have the form

$$S_\theta(f) \propto 1/f^2. \qquad (2.65)$$

For large t, $\Theta(t)$ is a Gaussian random variable. Indeed, the integrand $K(t) \triangleq \gamma(t+\Theta(t))N(t)$ in the RHS of (2.59) is a "weakly"-dependent stationary process. One may therefore invoke the central limit

[7]A Gaussian i.i.d. process is strictly speaking not well-defined, since it has infinite power (or variance). An arguably more precise formulation is the following: we suppose that the power spectrum of $N(t)$ is flat for all $f < f_c$ (with $f_c \gg 1/T_s$) and decays to zero for $f > f_c$, as for example in a Lorentzian spectrum with cut-off frequency f_c.

[8]The power spectral density of $\Theta(t)$ is not well-defined, since $\Theta(t)$ is not wide-sense stationary.

theorem (CLT) for such processes [126] [85] to conclude that $\Theta(t)$ is a Gaussian random variable.

Note that phase jitter does not alter the total power in the oscillator signal. If t is large and therefore $p_\Theta(\theta(t) = \theta) = \frac{1}{2\pi}$ for all θ, the power of the noisy signal is identical to the power of the periodic signal $x_s(t)$:

$$\mathrm{E}\big[x_s^2(t + \Theta(t))\big] = \frac{1}{T_s}\int_0^{T_s} x_s^2(t)dt. \tag{2.66}$$

The phase deviation $\Theta(t)$ does not change the total power, but it changes the *power spectral density*, potentially leading to effects as interchannel interference, as we pointed out earlier. If the noise sources are i.i.d. Gaussian random processes, the power spectral density has the shape of a Lorentzian around the carrier [54]; away from the carrier, the white-noise sources contribute a term that has a $1/f^2$ frequency dependence, and the colored-noise sources contribute terms that have a frequency dependence as $1/f^2$ multiplied with the spectral density of the colored-noise source [54]. In the case of flicker noise (FN), the most common colored-noise source, the dependency is therefore $1/f^3$. The power spectral density of the output signal $x(t)$ of a free-running clock with nominal frequency f_0 is usually of the form [110]:

$$S_x(f) = c/(f - f_0)^2 + c_{\mathrm{FN}}/|f - f_0|^3, \tag{2.67}$$

where the offset frequency $\Delta f \triangleq |f - f_0|$ is supposed to be sufficiently large, and c and c_{FN} are positive real numbers. A typical spectrogram of the *phase noise* $\theta(t)$ in free-running clocks has the shape [110]:

$$S_\theta(f) = d/f^2 + d_{FN}/f^3, \tag{2.68}$$

where d and d_{FN} are positive real numbers. Note that the $1/f^2$ behavior in the RHS of (2.68) is in agreement with (2.65).

From (2.59) we can also gain more insight about timing jitter. As we mentioned earlier, the effect of the phase deviation $\Theta(t)$ on a clock signal is to create jitter in the zero-crossing or transition times (see Fig. 2.12). Let us take one of the transitions of the signal as a time reference and let us synchronize it with $t = 0$. In an ideal signal, the transitions occur at $t = kT_s$ $(k = 1, 2, \ldots)$ as indicated by the arrows in Fig. 2.12. In a signal with phase deviation $\Theta(t)$, the transitions appear at $t = kT_s + \Theta_k$, where $\Theta_k \triangleq \Theta(kT_s)$. The expression (2.59) leads to the following model for the timing jitter Θ_k.

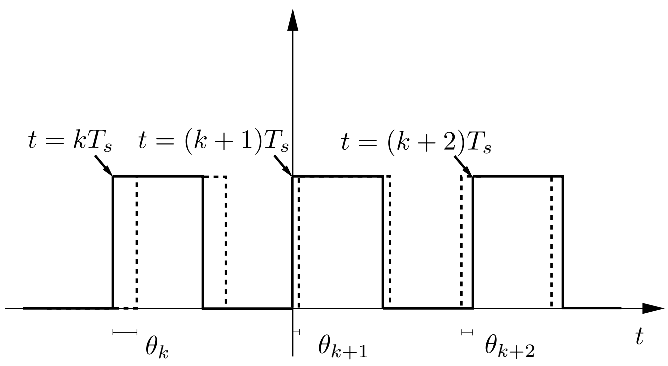


Figure 2.12: Timing jitter; ideal signal (solid line) and signal perturbed by phase noise (dashed line).

Discrete-time phase noise model:

$$\Theta_k = \Theta_{k-1} + N_k,$$

with

$$N_k \triangleq \int_{(k-1)T_s}^{kT_s} \gamma(t' + \Theta(t'))N(t')dt'. \tag{2.69}$$

The model holds generally for $\Theta_k \triangleq \Theta(k\Delta t)$, where Δt is an *arbitrary* positive real number (not necessarily equal to T_s, the period of $x_s(t)$).

We again consider the case where $N(t)$ is an *i.i.d. process* with (finite) variance σ_0^2. From (2.69) it follows that N_k is a zero-mean i.i.d. random process that is independent of Θ_k. The variance of N_k equals

$$\mathrm{Var}[N_k] = \mathrm{E}[N_k^2], \tag{2.70}$$

$$= \sigma_0^2 \int_{(k-1)T_s}^{kT_s} \mathrm{E}\big[\gamma^2(t' + \theta(t'))\big]dt'. \tag{2.71}$$

The integral in the RHS of (2.71) does not depend on k, hence the variance of N_k is constant:

$$\mathrm{Var}[N_k] \triangleq \sigma_N^2. \tag{2.72}$$

Note that N_k will be (close to) Gaussian as a consequence of the CLT, as we argued earlier [126] [85]. In summary, we derived the following model for timing jitter in a free-running clock perturbed by white noise sources.

(Discrete-time model for phase noise due to white-noise sources)

$$\Theta_k = \Theta_{k-1} + N_k, \tag{2.73}$$

with

$$N_k \sim \mathcal{N}_{0,\sigma_N^2}.$$

In words: the timing jitter Θ_k undergoes a Gaussian random walk process. The variance of Θ_k grows linearly with k, i.e.,

$$\mathrm{E}\big[\Theta_k^2\big] = \sigma_N^2\, k. \tag{2.74}$$

This is in agreement with experimental results: McNeill observed the linearly increasing variance for the timing of the transitions of a clock signal generated by an autonomous oscillator with white-noise sources [129]. On the other hand, if the oscillator also contains flicker-noise sources, the jitter variance initially grows linearly over time (due to the white noise), but after a sufficient amount of time, it increases *quadratically* (due to the flicker noise) [114].

Literature on Phase Noise in Free-Running Clocks

A large variety of phase models has been proposed in the literature.[9] Most of them are restricted to *particular* implementations, for example CMOS LC-oscillators [87]. The more general models are usually based on linear perturbation theory (see, e.g., [77] [104] [155]): the state $z(t)$ of the perturbed system is assumed to only slightly deviate from the state of the unperturbed system $x_s(t)$:

$$X(t) \triangleq x_s(t) + Z(t), \tag{2.75}$$

where $Z(t)$ is "small". The phase noise model is obtained by linearizing (2.47) around $x_s(t)$. Unfortunately, the expansion (2.75) is invalid

[9]We refer to [131, pp. 11–19] for a detailed overview of existing phase noise models.

for noisy oscillators, since $N(t)$ grows unboundedly in time due to phase drift [54]. Models derived from (2.75) are therefore rather problematic; for example, they predict the power of the oscillator signal to be infinite, which is clearly unphysical [54].

Hajimiri et al. [109] (see also [78]) analyzed phase noise starting from the *non-linear* expansion

$$X(t) \triangleq x_s(t + \Theta(t)) + Z(t), \tag{2.76}$$

where $\Theta(t)$ is the phase drift and $Z(t)$ is supposed to be small, as in (2.75). They propose the following model for $\Theta(t)$:

$$\Theta(t) = \int_{-\infty}^{t} \gamma(t')N(t')dt', \tag{2.77}$$

where $N(t)$ stands for noise sources in the oscillator. The difference between the models (2.59) and (2.77) is subtle: in (2.59), the function $\gamma(\cdot)$ depends on $t+\Theta(t)$, whereas in (2.77), $\gamma(\cdot)$ depends on t. Model (2.77) is simpler than (2.59), but unfortunately invalid.[10] It leads sometimes to incorrect results: for example, it is not capable of predicting injection locking, whereas model (2.59) accurately describes this phenomenon [198].

Demir et al. investigated phase noise by means of Floquet theory, which is a *non-linear* perturbation theory [54] [55] [57] [54] based on the expansion (2.76). The Floquet-analysis of [54] amounts to the model (2.59), which we derived previously in a rather intuitive manner. Demir et al. [55] determined how $\gamma(\cdot)$ depends on the function $f(\cdot)$ (cf. (2.59)): $\gamma(\cdot)$ turns out to be a so-called Floquet vector of $f(\cdot)$. In [57], also the case where $N(t)$ is *colored* stationary Gaussian noise is analyzed; it is proved that for large t, the marginal $p_\Theta(\theta(t))$ is a Gaussian pdf with constant mean and variance

$$\mathrm{Var}[\Theta(t)] \propto \int_{0}^{t} (t - t')R_N(t')dt', \tag{2.78}$$

where $R_N(\cdot)$ is the autocorrelation function of $N(t)$. In deriving (2.78), it is assumed that the bandwidth of $N(t)$ is much narrower than the frequency $\omega_s \triangleq 1/T_s$, or equivalently, it is assumed that the correlation width of the colored noise source in time is much larger than the oscillation period $T_s = 2\pi/\omega_s$. Demir et al. [54] [55] [57] formulated perhaps

[10]This can straightforwardly be verified by means of Fig. 2.11.

the most sophisticated and rigorous theory for phase noise currently available; nevertheless, many important questions remain unanswered. For example, in the case of colored-noise sources, the theory predicts that for large t the *marginal* $p_\Theta(\theta(t))$ is a Gaussian pdf, but it is unclear whether $\Theta(t)$ is a *jointly* Gaussian stochastic process. Generally speaking, the *dependency* between the phase drift $\Theta(t)$ at different time instances remains largely unspecified. In the following, we consider phase noise in phase-locked loops, which are oscillators that are driven by a reference signal.

2.4.2 Phase Noise in Phase-Locked Loops

Phase-locked loops (PLL) are widely used in passband receivers for acquiring and tracking the phase of the incoming carrier (cf. 2.2.2).[11] They contain a feedback loop (see Fig. 2.13) that tries to minimize the offset $\phi(t)$ between the phase of the local oscillator signal and the phase of the incoming signal, i.e.,

$$\phi(t) \triangleq \theta_R(t) - \theta_L(t), \tag{2.79}$$

where $\theta_R(t)$ and $\theta_L(t)$ is the phase of the incoming and local carrier respectively. Usually, the incoming carrier is generated by a free-running clock, and its phase $\theta_R(t)$ drifts as a random walk. A phase detector

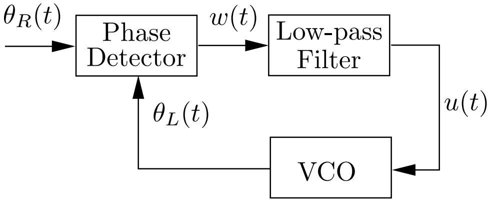


Figure 2.13: Phase-locked loop.

measures the phase offset $\phi(t)$; its output $w(t)$ typically depends on $\phi(t)$ in a non-linear fashion:

$$w(t) \triangleq f_{\mathrm{PD}}(\phi(t)). \tag{2.80}$$

[11]We refer to [135] for an in-depth discussion of PLLs.

The output of the phase detector $w(t)$ is passed through a low-pass filter, which is most often of first order. The resulting signal $u(t)$ is fed into a voltage-controlled oscillator (VCO), which generates a sinusoid whose phase $\theta_L(t)$ depends on $u(t)$ in the following way:

$$\theta_L(t) = \gamma_u \int_{-\infty}^{t} u(t')dt', \tag{2.81}$$

where γ_u is a positive real number.[12] In other words, the VCO tries to decrease the phase error $\phi(t)$ by adjusting the phase $\theta_L(t)$ of the local carrier according to the low-pass filtered error signal.

The non-linearity in the phase detector is responsible for the rather complex behavior of PLLs. A typical realization of the phase error $\phi(t)$ is shown in Fig. 2.14. When the PLL is switched on, it has no information about the phase θ_R. The initial phase error is therefore large, but it decreases steadily in a transient regime called *acquisition*. After some time, the phase error has become small and the PLL is said to be "locked" unto the incoming reference. Occasionally, this quasi-stationary regime may be interrupted: due to some random disturbance, the PLL may become unstable during a short amount of time, and then lock back unto θ_L. Or, much worse, it may lock unto some neighboring stationary point $\theta_L + k2\pi/M \quad (k \in \mathbf{Z})$, where M is the alphabet size of the data symbols modulated on the incoming carrier. This phenomenon, called *cycle slip*, causes large phase errors, and can drastically increase the bit (or word) error rates. Cycle slips occur with low probability in practical systems, but they are unavoidable, especially at low SNR.

The non-linearity of the phase detector seriously complicates the analysis of PLLs. As a consequence, few (exact) analytical results on phase noise in PLLs are available. Mehrotra [132] proposed a rigorous mathematical model for PLLs, similar in spirit as the Floquet analysis of Demir et al. [54] [57] for free-running oscillators. The approximate expression (2.83) is replaced by

$$\Theta_L(t) = \int_{-\infty}^{t} \Big[\gamma_u\big(t' + \Theta_L(t')\big)U(t') + \gamma_n\big(t' + \Theta_L(t')\big) \cdot M(t')\Big]dt', \tag{2.82}$$

where $M(t)$ stands for the internal noise sources of the VCO. The expression (2.82) is a natural extension of the model (2.59); it is more a

[12] In the expression (2.83), the noise sources in the VCO are neglected; we will come back to this issue later on.

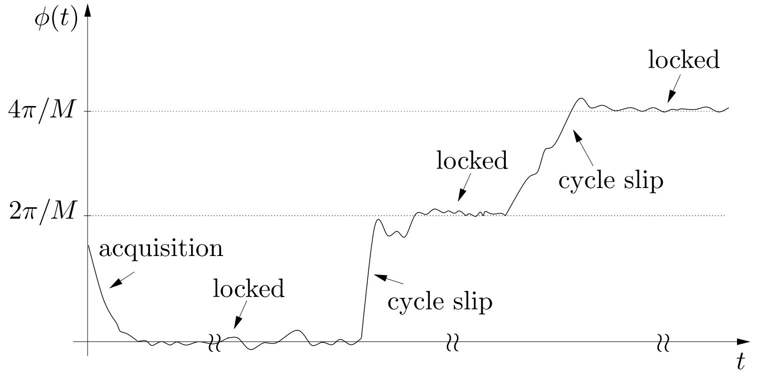


Figure 2.14: Phase error in a PLL.

precise description of the dynamics of a VCO than (2.83). Unfortunately, the resulting stochastic differential equations for the phase error $\phi(t)$ are hard to solve. Mehrotra only obtained (partial) results for PLL with *white* noise sources. We refer to [132] for more details.

It is more common to model PLL by means of nonlinear differential equations with *additive* noise sources. The expression (2.82) is approximated by

$$\Theta_L(t) = M(t) + \gamma_u \int_{-\infty}^{t} U(t')dt', \tag{2.83}$$

where the stochastic process $M(t)$ models the phase noise in the VCO. The incoming signal is assumed to be a perfect periodic signal with additive noise, i.e., the phase drift of the incoming signal is *not* taken into account. It has been shown that, under the additional assumption of *white* noise sources, the stationary distribution of the phase noise in a *locked* PLL *without* low-pass filter ("first-order PLL") is a Tikhonov distribution [202, Chapter 4]:

$$p_\Phi(\phi) = \frac{\exp(\alpha_s \cos\phi)}{2\pi I_0(\alpha_s)}, \quad (|\phi| < \pi) \tag{2.84}$$

where I_0 is the zeroth-order Bessel function of the first kind, and α_s is a positive real number. No (exact) analytical results seem to exist for PLLs *with* low-pass filters.[13] Nevertheless, the distribution (2.84) seems

[13]PLLs with first-order low-pass filters ("second-order PLLs") are most often used

to be a good approximation of experimentally obtained histograms of the phase error in such PLLs [202, pp. 112-117]. Note that the Tikhonov distribution can be well approximated by a Gaussian distribution if α_s is sufficiently large.

A popular approximation method [135, Chapter 2 and 3] is to *linearize* the phase detector characteristic $f_{\mathrm{PD}}(\phi)$, which is only valid as long as the phase error $\phi(t)$ remains small. The PLL is described by *linear* differential equations with *additive* noise sources, which can be solved by standard linear systems theory [135, pp. 48–53] (see also [110] and [132]). A key result from this analysis is that the spectrum of the phase noise is given by

$$S_\phi(s) = |H(s)|^2 S_N(s) + |1 - H(s)|^2 S_M(s), \tag{2.85}$$

where $S_N(s)$ and $S_M(s)$ is the power spectral density of the additive noise in the received signal and VCO respectively, and $H(s) \triangleq \frac{\theta_L(s)}{\theta_R(s)}$ is the closed-loop transfer function, which is a low-pass filter. As a consequence of (2.85), the variance of the phase error $\mathrm{Var}[\phi(t)]$ converges to the (finite) value:

$$\begin{aligned} \mathrm{Var}[\phi(t)] &= \frac{1}{\pi}\int_0^\infty S_\phi(j\omega)d\omega & (2.86) \\ &= \frac{1}{\pi}\int_0^\infty \Big[|H(j\omega)|^2 S_N(j\omega) + |1 - H(j\omega)|^2 S_M(j\omega)\Big] d\omega. & (2.87) \end{aligned}$$

The expression (2.85) indicates that, under "normal" circumstances, the phase error in a PLL is bounded, in contrast to the situation in free-running clocks. It can also be seen from (2.85) that in order to minimize the phase noise $\phi(t)$ due to $N(t)$, the additive noise in the received signal, the bandwidth of $H(s)$ should be made as *narrow* as possible. On the other hand, the bandwidth should be chosen *wide* in order to avoid the phase noise due the VCO instabilities $M(t)$. Both criteria are opposed to one another, and one needs to make a compromise. It is common practice to tune the parameters of the low-pass filter such that the steady-state phase error variance (2.87) is as small as possible [135, pp. 150–153]. We underline once more that the expression (2.87) only holds as long as the PLL is locked and the phase error is small; the behavior of the PLL with regard to cycle slips is also often taken into account in the PLL-design process.

in practice.

2.4.3 Heuristic Models for Phase Noise

As we have seen previously, a rigorous (tractable) model for phase noise is currently not available. However, for practical purposes, heuristic models may suffice. The phase jitter in PLLs is sometimes approximated by low-pass Gaussian noise. An example of such a model is the following first-order autoregressive (AR) process [108].

(First-order AR model for phase noise in a PLL [108])

$$\Theta_k = a\,\Theta_{k-1} + N_k, \tag{2.88}$$

where

$$N_k \sim \mathcal{N}_{0,\sigma_N^2},$$

and $a < 1$.

The phase fluctuations in a PLL usually also contain high-pass components (cf. (2.85)). They are neglected in the above model. We propose the following higher-order autoregressive-moving average (ARMA) process as a more accurate model for phase noise in PLLs.

(Higher-order ARMA model for phase noise in a PLL)

$$\begin{aligned}
\Psi_k &= \sum_{\ell=0}^{L_a} a_\ell N_{k-\ell} + \sum_{\ell=0}^{L_b} b_\ell \Psi_{k-\ell} && (2.89)\\
\Phi_k &= \Phi_{k-1} + Z_k && (2.90)\\
\Upsilon_k &= \sum_{\ell=0}^{L_a} a_\ell \Phi_{k-\ell} + \sum_{\ell=0}^{L_b} b_\ell \Upsilon_{k-\ell} && (2.91)\\
\Xi_k &= \Phi_k - \Upsilon_k && (2.92)\\
\Theta_k &= \Psi_k + \Xi_k, && (2.93)
\end{aligned}$$

where

$$\begin{aligned}
N_k &\sim \mathcal{N}_{0,\sigma_N^2}, && (2.94)\\
Z_k &\sim \mathcal{N}_{0,\sigma_Z^2}. && (2.95)
\end{aligned}$$

The process Ψ_k stands for the phase noise induced by the additive white noise in the received signal (cf. the first term in (2.85)). It is generated by passing the white noise process N_k through the low-pass filter

$$H(z) \triangleq \frac{\sum_{\ell=0}^{L_a} a_\ell z^k}{\sum_{\ell=0}^{L_b} b_k z^k}. \tag{2.96}$$

The order of $H(z)$ is given by the order of the low-pass filter in the PLL. For example, to model first-order PLLs, which are the most widely used PLLs, ARMA-processes of second-order (i.e., $L_a = L_b = 2$) are the most suitable. The coefficients a_k and b_k can be determined by measurements of the phase jitter. The random-walk phase drift of the VCO is modeled by (2.90). The process Υ_k stands for the low-pass filtered VCO phase noise, generated by passing Φ_k through $H(z)$. The high-pass filtered VCO phase noise (cf. the second term in (2.85)) is modeled by Ξ_k. It is obtained by filtering Φ_k by $1 - H(z)$ (cf. (2.91) and (2.92)). The total phase noise in the PLL is represented by Θ_k, the sum of the low-pass phase noise Ψ_k and the high-pass phase noise Ξ_k.

The model does not incorporate flicker noise, since it is hard (if not impossible) to represent flicker noise as an ARMA process (cf. (2.4)). However, this is not really problematic. Only the high-pass components of the VCO phase noise contribute to the PLL phase noise (cf. the second term in (2.85)), hence the VCO's flicker-noise sources are usually less critical than its white-noise sources.

The model also disregards cycle slips, it only describes *locked* PLLs. By combining it with a finite state machine that models cycle slips, a more complete (heuristic) description for phase noise in a PLL can be obtained.

The random-walk process (2.73) (or equivalently (2.88), with $a = 1$) is widely used as a model for phase noise in free-running clocks. We will mostly use that model in this thesis. Note that the random-walk phase model is valid if the oscillator only contains white-noise sources. It should be straightforward to extend the results of this thesis to the (more realistic) ARMA-model (2.89)–(2.93).

2.5 Random-walk Phase Model

We extend the model (2.22)–(2.23) with random phase fluctuations.

Random-walk phase model:

$$Y_k = X_k e^{j\Theta} + N_k, \tag{2.97}$$

where

$$\Theta_k = (\Theta_{k-1} + W_k) \bmod 2\pi, \tag{2.98}$$

and

$$\begin{aligned} N_k &\sim \mathcal{N}_{0,\sigma_N^2}, & (2.99) \\ W_k &\sim \mathcal{N}_{0,\sigma_W^2}. & (2.100) \end{aligned}$$

The model describes the received symbols y_k accurately if

a) a timing synchronizer tracks the timing offsets and the variance σ_W^2 is small (e.g. $\sigma_W < 10^{-2}$); the expressions (2.21) and (2.21) are then valid

b) the frequency offset $\nu = 0$,

c) the fluctuations of the phase offset $\theta \triangleq \theta_L - \theta_R - 2\pi f_R \tau$ can be described as a random walk process.

The last assumption is for example met if the oscillators in the transmitter and receiver are free-running clocks (perturbed by white-noise sources), the frequency f_R is approximately constant, and the relative distance between the transmitter and receiver (and hence the channel delay τ) is constant or fluctuates as a random walk.

Chapter 3

Factor Graphs and Summary Propagation

This chapter aims at giving an introduction to factor graphs and the summary-propagation algorithm on a generic level. Factor graphs are defined in Section 3.1 as graphs (in the mathematical sense) that represent the factorization of multivariate functions. One of the most important operations that can be performed on factor graphs is marginalization, i.e., the computation of marginals of probability functions. Marginalization lies at the heart of many algorithms in signal processing, coding and machine learning. As we will show, computing marginals amounts to passing messages ("summaries") along the edges in the factor graph of the system at hand. This generic message-passing algorithm, called the sum(mary)-product algorithm (SPA), is introduced in Section 3.2.

Literature

In the present and next chapter, we closely follow Loeliger's tutorial introduction about factor graphs and the summary-propagation algorithm [119].

Factor graphs have their roots in coding theory; they were originally introduced in [103], based on earlier ideas by Tanner [190] and Wiberg et al. [211] [210]. In this thesis, we use the refined notation proposed by Forney [66] (there called "normal graphs"). We will refer to this notation as Forney-style factor graphs, or short FFG.

Since the summary-product algorithm is very generic, it has repeatedly been re-discovered in different scientific communities. The algorithm has a long history in the context of error correcting codes: Gallager's algorithm for decoding low-density parity check (LDPC) codes [72] [73], the BCJR-algorithm [15] and the Viterbi algorithm [65] can be regarded as early versions of the summary-product algorithm. In the statistics and machine-learning community, the algorithm is known under the name of "probability propagation" or "belief propagation" [157] [68] and is usually applied on Bayesian networks. Yet another instance of the same principle is the Bethe and Kickuchi method from statistical physics (see [223] and references therein). Two standard techniques in signal processing, i.e., hidden-Markov models and Kalman filtering, can also be seen as instances of the summary-product algorithm. For a more detailed review of the history of the summary-product algorithm within different scientific fields, we refer to [103] [222].

3.1 Factor Graphs

Factor graphs belong to the family of graphical models. A graphical model is—as the name suggests—a graphical representation of some mathematical model; it visualizes interactions between variables in a model. Examples of such models are error-correcting codes, mathematical descriptions of communications channels, input-output systems as for example in classical filter and control theory, electrical networks, spin glasses (which are physical models of magnetic materials), ordinary or partial differential equations, and statistical models of speech signals, images or video sequences. There are various types of graphical models besides factor graphs, most importantly: Markov random fields [213] (which are often used in machine vision, statistics and statistical physics), Bayesian networks [157], and neural networks[1] [22]. The latter two originated in the machine learning community, but have meanwhile found their way to research fields as diverse as microbiology [80],

[1]We will encounter a specific type of neural network, the so-called feed-forward neural network, in Appendix E.

particle physics [79] and geophysics [174].

Remark 3.1. (Advantages of factor graphs)
We decided to use factor graphs for the following reasons [119]:

- They allow hierarchical modeling ("boxes within boxes").[2]
- They are compatible with standard block diagrams (see, e.g., Appendix H).
- The summary-product message update rule can most elegantly be formulated on factor graphs, especially on Forney-style factor graphs.

As already mentioned before, factor graphs represent functions. Let us have a look at some first examples.

Example 3.1. (Factor graph of a function without structure)
The factor graph of the function $f(x_1, x_2, x_3)$ is shown in Fig. 3.1 (left): edges represent variables, and nodes represent factors. An edge is connected to a node if and only if the corresponding variable is an argument of the corresponding function. $\square$

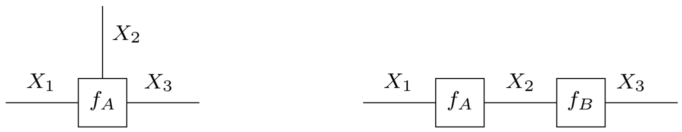


Figure 3.1: Factor graph of function without structure (right) and a function with structure (left).

The concept of factor graphs becomes interesting as soon as the function to be represented has structure, i.e., when it factors.

[2]This feature is for example also available in commercial CAD-packages for architectural or VLSI design. The operation of "closing a box", i.e., the marginalization of the variables inside a box (subgraph), is at the core of statistical physics, whose aim is to understand how systems of many interacting parts can display high-level simplicity (e.g., the kinetic theory of gasses). We refer to [188] for an elaborated philosophical account on this issue.

Example 3.2. (Factor graph of a function with structure)
Let us assume that the function $f(x_1, x_2, x_3)$ of Example 3.1 factors as $f(x_1, x_2, x_3) \triangleq f(x_1, x_2) f(x_2, x_3)$; the factor graph of Fig. 3.1 (right) represents this factorization. We call f the *global* function and f_1 and f_2 *local* functions. $\square$

Example 3.3. The (global) function

$$f(x_1, x_2, x_3, x_4, x_5, x_6) \triangleq f_A(x_1, x_2) f_B(x_3, x_4) f_C(x_2, x_4, x_5) f_D(x_5, x_6) \tag{3.1}$$

is represented by the factor graph in Fig. 3.2 $\square$

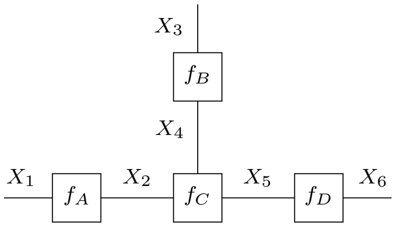


Figure 3.2: An example factor graph.

More formally, a Forney-style factor graph (FFG) is defined as follows:

- **Factor graph:** An FFG represents a function f and consists of nodes and edges. We assume that f can be written as a product of factors.
- **Global functions:** The function f is called the global function.
- **Nodes/local functions:** There is a node for every factor, also called local function.
- **Edges/variables:** There is an edge or half-edge for every variable.
- **Connections:** An edge (or half-edge) representing some variable X is connected to a node representing some factor f if and only if f is a function of X.
- **Configuration:** A configuration is a particular assignment of values to all variables. We use capital letters for unknown variables

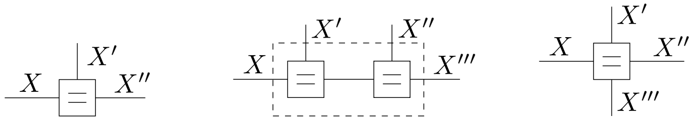


Figure 3.4: Equality constraint node used for variable replication. (left) single node; (right) compound node; (middle) the compound node as concatenation of single nodes.

and small letters both for particular values of such variables[3] and for known (observed) variables.

- **Configuration space:** The configuration space Ω is the set of all configurations: it is the domain of the global function f. One may regard the variables as functions of the configuration ω, just as we would with random/chance variables.

- **Valid configuration:** A configuration $\omega \in \Omega$ will be called valid if $f(\omega) \neq 0$.

Remark 3.2. (Cloning variables)
Implicit in the previous definition is the assumption that no more than two edges are connected to one node. This restriction is easily circumvented by introducing variable replication nodes (also referred to as "equality constraint nodes"). An equality constraint node represents the factorization $\delta(x - x')\delta(x' - x'')$, and is depicted in Fig. 3.4 (left). It enforces the equality of the variables X, X', and X''.[4] The (single) equality constraint node generates two replicas of X, i.e., X' and X''. If more replicas are required, one can concatenate single nodes as shown in Fig. 3.4 (middle); combining those single nodes ("boxing") leads to a compound equality constraint node (see Fig. 3.4 (right)).

Most graphical model are associated to a particular algorithm. For example, feed-forward neural networks and the back-propagation algorithm go hand in hand [22]; Markov random fields have strong ties to Gibbs sampling. Traditionally, factor graphs are associated to the sum(mary)-product algorithm. Throughout this thesis, however, it will become clear

[3]This imitates the notation used in probability theory to denote chance/random variables and realizations thereof.

[4]The factor graph notation of [103] is obtained by replacing each equality constraint node by a circle.

that many other algorithms can conveniently be described by means of factor graphs.[5]

3.2 Summary-Propagation Algorithm

In a wide variety of problems (e.g., in signal processing, coding, estimation, statistical physics, and machine learning), one is interested in marginal probabilities of certain variables in the system. We explain now how such marginals can be computed by message passing on a factor graph.

3.2.1 Summary Propagation on Factor Trees

Example 3.4. (Marginalization of a factored function)
Let us consider again the global function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ of Example 3.3. Suppose we are interested in the marginal function

$$f(x_5) \quad \triangleq \sum_{x_1,x_2,x_3,x_4,x_6} f(x_1, x_2, x_3, x_4, x_5, x_6). \tag{3.2}$$

With the factorization (3.1), we have:

$$\begin{aligned} f(x_5) &= \sum_{x_1,x_2,x_3,x_4,x_6} f_A(x_1,x_2) \cdot f_B(x_3,x_4) \cdot f_C(x_2,x_4,x_5) \cdot f_D(x_5,x_6) \\ &= \underbrace{\sum_{x_2,x_4} f_C(x_2,x_4,x_5) \underbrace{\left(\sum_{x_1} f_A(x_1,x_2)\right)}_{\mu_{f_A\to x_2}(x_2)} \cdot \underbrace{\left(\sum_{x_3} f_B(x_3,x_4)\right)}_{\mu_{f_B\to x_4}(x_4)} \cdot}_{\mu_{f_C\to x_5}(x_5)} \\ &\quad \underbrace{\left(\sum_{x_6} f_D(x_5,x_6)\right)}_{\mu_{f_D\to x_5}(x_5)}. \end{aligned} \tag{3.3}$$

□

[5]Later on, we will clarify how both the back-propagation algorithm and Gibbs sampling for example can be viewed as summary-propagation on factor graphs.

The idea behind (3.3) is to "push" the summations as much right as possible. For example, when summing w.r.t. X_6, we can push the summation sign to the right side of every factor except $f_D(x_5, x_6)$, since this factor depends on X_6. As a result, instead of carrying out a high-dimensional sum, it suffices to carry out simpler ones (one- and two-dimensional in our example). The intermediate terms $\mu_{f_j \to x_i}(x_i)$ are functions of X_i. The domain of such a functions is the alphabet of X_i. Their meaning becomes obvious when looking at Fig. 3.5.

The intermediate results can be interpreted as "messages" flowing along the edges of the graph. For example, the message $\mu_{f_A \to x_2}(x_2)$, which is the sum $\sum_{x_1} f_A(x_1, x_2)$, can be interpreted as a message leaving node f_A along edge X_2. If both $\mu_{f_A \to x_2}(x_2)$ and $\mu_{f_B \to x_4}(x_4)$ are available, the message $\mu_{f_C \to x_5}(x_5)$ can be computed as the output message of node f_C towards edge X_5. The final result of (3.3) is

$$f(x_5) = \mu_{f_C \to x_5}(x_5) \cdot \mu_{f_D \to x_5}(x_5). \tag{3.4}$$

It is the product of the two messages along the same edge.

Each message can be regarded as a "summary" of what lies "behind" it, as illustrated by the boxes in Fig. 3.5. Computing a message means "closing" a part of the graph ("box"). The details inside such a box are "summed out", only a summary is propagated (hence the name summary-propagation). In the first step, the dark shaded areas in Fig. 3.5 are boxed (resulting in $\mu_{f_A \to x_2}(x_2)$ and $\mu_{f_D \to x_5}(x_5)$). Afterwards, the lighter shaded box is closed (amounting to $\mu_{f_C \to x_2}(x_2)$), until we arrive at (3.4).

Half-edges (such as X_1) do not carry a message towards the connected node; alternatively, the edge may be thought of as carrying a message representing a neutral factor 1. With this in mind, we notice that every message (i.e., every intermediate result) of (3.3) is computed in the same way. Consider the generic node depicted in Fig. 3.8 with messages arriving along its edges $X_1, \ldots, X_N$. The message towards edge y is computed by the following rule.

Sum-product rule:

$$\begin{aligned} \mu_{f \to y}(y) \quad &\triangleq \quad \sum_{x_1, \ldots, x_N} f(y, x_1, \ldots, x_N) \cdot \\ &\qquad\qquad \mu_{x_1 \to f}(x_1) \cdots \mu_{x_N \to f}(x_N) \end{aligned} \tag{3.5}$$

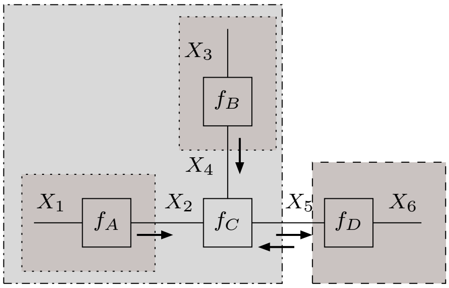


Figure 3.5: Summary-propagation for computing $f(x_5)$.

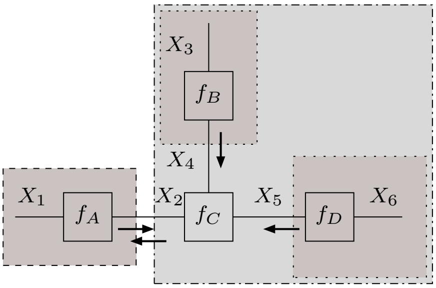


Figure 3.6: Summary-propagation for computing $f(x_2)$.

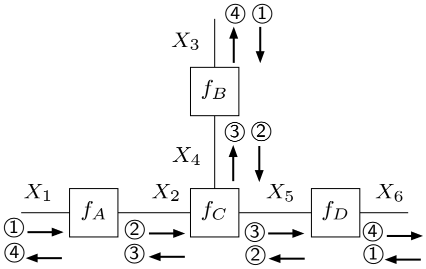


Figure 3.7: The SPA computes two messages along each edge. Those messages are required for calculating the marginal functions $f(x_1)$, $f(x_2)$, $f(x_3)$, $f(x_4)$, $f(x_5)$ and $f(x_6)$. The circled numbers indicate the order of the message computations.

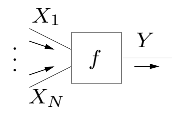


Figure 3.8: Message along a generic edge.

In words: The message out of a node f along the edge Y is the product of the function f and all messages towards f along all other edges, summarized over all variables except Y. This is the **sum-product rule**. In general, messages are computed out of any edge, there is no preferential direction. The message out of a leaf node f along edge Y is the function f itself, as illustrated in Fig. 3.9.

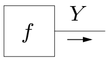


Figure 3.9: Message out of a leaf node.

The sums in (3.2), (3.3) and (3.5) can be replaced by any "summary operator", e.g., the integral operator for continuous-valued variables or the max operator for performing maximizations; this leads to the **integral-product rule** and **max-product rule** respectively.

Integral-product rule:

$$\mu_{f\to y}(y) \quad \triangleq \quad \int \underset{\mathcal{D}}{\cdots} \int f(y, x_1, \ldots, x_N) \cdot \mu_{x_1\to f}(x_1) \cdots \mu_{x_N\to f}(x_N)\, dx_1 \cdots dx_N \qquad (3.6)$$

Max-product rule:

$$\mu_{f\to y}(y) \quad \triangleq \quad \max_{x_1,\ldots,x_N} f(y, x_1, \ldots, x_N) \cdot \mu_{x_1\to f}(x_1) \cdots \mu_{x_N\to f}(x_N) \qquad (3.7)$$

The above rules can be considered as instances of the following single rule.

Summary-product rule: The message $\mu_{f \to y}(y)$ out of a factor node $f(y, \ldots)$ along the edge Y is the product of $f(y, \ldots)$ and all messages towards f along all edges except Y, summarized over all variables except Y.

We will encounter other examples of this general rule in Chapter 4. In the same chapter, we will also consider *different* rules (i.e., the E-log rule and extensions, cf. Section 4.9).

The following example shows how several marginals can be obtained simultaneously in an efficient manner.

Example 3.5. (Recycling messages)
Suppose we are also interested in the marginal function $f(x_2)$ of the global function $f(x_1, x_2, x_3, x_4, x_5, x_6)$ of Example 3.3:

$$f(x_2) \triangleq \sum_{x_1, x_3, x_4, x_5, x_6} f(x_1, x_3, x_4, x_5, x_6).$$

This marginal can be computed by the summary-propagation depicted in Fig. 3.6. Note that we have already computed the messages $\mu_{f_A \to x_2}(x_2)$, $\mu_{f_B \to x_4}(x_4)$, and $\mu_{f_D \to x_5}(x_5)$ in (3.3); they can be "re-used" for computing $f(x_2)$. Eventually, $f(x_2)$ is obtained as

$$f(x_2) = \mu_{f_A \to x_2}(x_2) \mu_{f_C \to x_2}(x_2). \tag{3.8}$$

$\square$

From this last example, we learn that the two messages associated to an edge are for the computation of each marginal the same. It is therefore sufficient to compute each message once. The marginal $f(y)$ of a certain variable Y is the product of the two messages on the corresponding edge, such as (3.4) and (3.8). In general, it is

$$\boxed{f(y) = \mu_{f_A \to y}(y) \cdot \mu_{f_B \to y}(y)} \tag{3.9}$$

where f_A and f_B are the two nodes attached to edge Y. For half edges, the message coming from the open end carries a neutral factor "1". Therefore, the message from the node towards the edge is already the marginal of the corresponding variable.

In its general form, the **summary-propagation algorithm (SPA)** computes two messages on every edge. For factor graphs without loops (factor trees), the marginals can obtained in an optimal number of computations as follows.[6] One starts the message computation from the leaves and proceeds with nodes whose input messages become available. In this way, each message is computed exactly once, as illustrated in Fig. 3.7. When the algorithm stops, exact marginals, such as (3.9), are available for all variables *simultaneously*.

In summary:

- Marginals such as (3.2) can be computed as the product of two messages as in (3.9).
- Such messages are summaries of the subgraph behind them.
- All messages (except those out of terminal nodes) are computed from other messages according to the summary-product rule.

Remark 3.3. (Scaling)
If one applies the rules (3.5) (or (3.6) and (3.7)), the values of the messages often quickly tend to zero and the algorithm becomes instable. Therefore, it is advisable to scale the message: instead of the message $\mu(.)$, a modified message $\tilde{\mu}(.) \stackrel{\triangle}{=} \gamma\mu(.)$ is computed, where the scale factor γ may be chosen as one wishes. The final result (3.9) will then be known up to a scaling factor, which is often not a problem. In some applications, however, the scaling factors are of central importance (see Chapter 6).

Remark 3.4. (Message update schedule)
A message update schedule says when one has to calculate what message. For factor trees, there is an optimal message update schedule, as we explained previously; for cyclic factor graphs, this is not the case.

3.2.2 Summary Propagation on Cyclic Factor Graphs

The situation becomes quite different when the graph has cycles. In this case, the summary-propagation algorithm becomes iterative: a new

[6]The number of computations may be reduced by additional information about the structure of the local node functions. This is the case when the factor nodes themselves may be expressed by (non-trivial) factor trees.

output message at some node can influence the inputs of the same node through another path in the graph. The algorithm does not amount to the exact marginal functions. In fact, there is even no guarantee that the algorithm converges! Astonishingly, applying the summary-product algorithm on cyclic graphs works excellently in the context of coding and signal processing, and machine learning. In many practical cases, the algorithm reaches a stable point and the obtained marginal functions are satisfactory: decisions based on those marginals are often close enough to the "optimal" decisions.

Summary-propagation on cyclic-graphs consists of the following steps

a) First, all edges are initialized with a neutral message, i.e., a factor $\mu(.) = 1$.

b) All messages are then recursively updated according to some schedule. This schedule may vary from step to step.[7]

c) After each step, the marginal functions are computed according to (3.9).

d) One takes decisions based on the current marginal functions.

e) The algorithm is halted when the available time is over or when some stopping criterion is satisfied (e.g., when all messages varied less than some small ε over the last iterations).

Remark 3.5. (Understanding summary propagation on cyclic graphs)
The theoretical understanding of summary propagation on cyclic graphs is sparse. Some particular results are available, e.g., for factor graphs with one loop [5] or, e.g., factor graphs representing jointly Gaussian densities [209] [173]. Yedidia et al. have shown that the fixed-points of the sum-product algorithm are in a one-to-one relationship with zero-gradient points of a Bethe free energy associated to the underlying problem [223] (see also [127]). This result was later refined by Heskes, who proved that the stable fixed-points of loopy belief propagation (i.e., iterative sum-product algorithm) are in fact *minima* of the Bethe free energy [83]. These insights paved the road to alternative message-passing algorithms such as generalized belief propagation [223], convex-concave-procedure algorithms (CCCP) [225], and "fractional" (or "convexified") belief propagation [212] [205].

[7]For more details on scheduling, see [103].

3.3 Factor-graph Transformation: Clustering

In this section, we describe how the structure of a factor graph can straightforwardly be modified by a technique called clustering [103]. For alternative transformations such as stretching and the junction-tree method, we refer to [103] and [6]. Let us have a look at the following example.

Example 3.6. We consider the (cyclic) factor graph of Fig. 3.10(a) representing

$$f(x,y,z) \triangleq f_A(x) f_B(x,y) f_C(x,z) f_D(y) f_E(y,z) f_F(z). \tag{3.10}$$

By clustering the edges Y and Z and connecting the neighboring nodes of both edges to the new clustered node, we obtain the factor graph shown in Fig. 3.10(b); it represents the factorization

$$f'(x,y,z) \triangleq f_A(x) f'_B(x,y,z) f'_C(x,y,z) f'_D(y,z) f'_E(y,z) f'_F(y,z). \tag{3.11}$$

The node f_E connecting Y and Z in the original factor graph appears with just a single edge in the new factor graph. Note also that there are two local nodes connecting X to (Y,Z), i.e., also the new factor graph is cyclic. The local nodes in the new factor graph retain their dependencies from the original factor graph. For example, $f'_C(x,y,z)$ is connected to X and the pair (Y,Z), but it actually does not depend on Z, hence $f'_C(x,y,z) \triangleq f_C(x,z)$. Similarly, $f'_B(x,y,z) \triangleq f_B(x,y)$, $f'_C(x,y,z) \triangleq f_C(x,z)$, $f_D(y,z) \triangleq f_D(y)$, $f'_E(y,z) \triangleq f_E(y,z)$ and $f'_F(y,z) \triangleq f_F(z)$, and therefore $f'(x,y,z) \triangleq f(x,y,z)$; both the original and the new factor graph represent the same global function.

The cycle in the factor graph of Fig. 3.10(b) can be removed by clustering the nodes $f'_B(x,y,z)$ and $f'_C(x,y,z)$

$$f'_{BC}(x,y,z) \triangleq f'_B(x,y,z) f'_C(x,y,z), \tag{3.12}$$

resulting in the factor graph of Fig. 3.10(c); the new global function is

$$f'(x,y,z) \triangleq f_A(x) f'_{BC}(x,y,z) f'_D(y,z) f'_E(y,z) f'_F(y,z), \tag{3.13}$$

which is identical to the original global function (3.10). □

Remark 3.6. (Removing cycles by clustering)
In the previous example, we have removed a cycle from the factor graph by clustering edges and nodes. The resulting factor graph is cycle-free and the sum-product algorithm may be applied to compute *exact* marginals. The *complexity* of the messages in the graph, however, has increased. Let Y and Z have alphabets A_y and A_z respectively; the alphabet of the pair (Y, Z) is $A_y \times A_z$. The domain size of (Y, Z) is equal to the *product* $|A_y||A_z|$, where $|A_y|$ and $|A_z|$ denote the size of the alphabets A_y and A_z respectively. Therefore, if the messages are represented by a list of their values (as is common practice when Y and Z take values in a finite set), the length of the messages of (X, Y) (e.g., the message from f_C to (X, Y)) is also equal to $|A_y||A_z|$. This can imply a substantial cost increase in computational complexity of the sum-product algorithm. If the messages are represented in alternative ways however, as for example by Gaussian distributions, the complexity does not necessarily increase exponentially; clustering may then be a practical solution to handle cycles. We will demonstrate this idea in Section 5.3.7.

We end this chapter by formulating the general procedure to cluster edges (variables) and nodes (factors) in a factor graph.

Clustering nodes: Nodes $f_1, f_2, \ldots, f_n$ are clustered as follows

a) Delete $f_1, f_2, \ldots, f_n$ and any incident edge from the factor graph,

b) Introduce a new node representing the n-tuple $(f_1, f_2, \ldots, f_n)$,

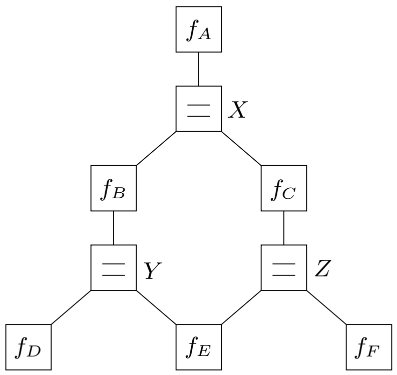
c) Connect this new node to nodes that were neighbors of $f_1, f_2, \ldots, f_n$ in the original graph.

Clustering edges: Edges $X_1, X_2, \ldots, X_n$ are clustered as follows

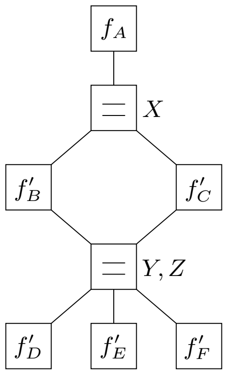
a) Delete $X_1, X_2, \ldots, X_n$ and any incident node from the factor graph,

b) Introduce a replication node representing the n-tuple $(X_1, X_2, \ldots, X_n)$,

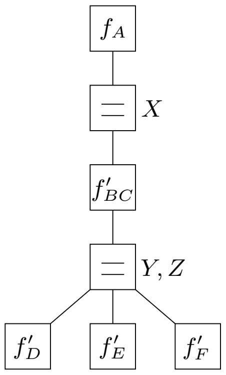
c) Connect this new node to nodes that were neighbors of $X_1, X_2, \ldots, X_n$ in the original graph.



(a) Original factor graph.



(b) Clustering Y and z.



(c) Clustering f_B and f_C.

Figure 3.10: Clustering transformation.

Chapter 4

Phase-Estimation Algorithms

In this chapter, we develop code-aided carrier-phase-estimation algorithms for single-carrier communications systems. In contrast to earlier and parallel work, we mainly focus on how such algorithms can be derived in a systematic fashion from the factor graph of the system at hand. The starting point of our approach is to apply the sum-product algorithm on that factor graph, which leads to intractable integrals. We consider several methods to approximate those integrals; each method corresponds to a certain message type and leads to a different phase-estimation algorithm. We consider the following approximation methods:

- numerical integration,
- particle methods,
- adaptive quantization,
- gradient methods,
- expectation maximization (EM).

Before we apply each method to the *particular* application of carrier-phase estimation, we will outline how each approximation method *in* *general* can be viewed as message passing on factor graphs. We will also consider well-known instances of some of the methods (e.g., particle methods such as Gibbs sampling, MCMC etc.).

The Appendices D and E are heavily based on material presented in this chapter. In Appendix D, we outline how kernels can be derived from graphical models, in particular, by the message-passing methods we present in this chapter. In Appendix E, we demonstrate how the back-propagation algorithm for the training of feed-forward neural networks can be derived as message-passing in factor graphs.

This chapter may be of interest to readers who want to learn more about:

- code-aided carrier-phase estimation, or, code-aided **channel estimation** in general;
- **message-passing algorithms** for estimation with applications to model-based signal processing and machine learning.

The results on phase estimation we present in this chapter are based on [47] [52] [48] [44]. The message-passing viewpoint presented in this chapter was developed in collaboration with Sascha Korl.

4.1 Introduction

We consider channels of the form

$$Y_k = X_k e^{j\Theta_k} + N_k, \tag{4.1}$$

where X_k is the channel input symbol at time $k \in \{1, 2, \ldots, L\}$, Y_k is the corresponding received symbol, $\Theta_k \in [0, 2\pi)$ is the unknown phase, and N_k is complex white Gaussian noise with (known) variance $2\sigma_N^2$, i.e., σ_N^2 per dimension. We will use the notation $X \triangleq (X_1, X_2, \ldots, X_L)$, $Y \triangleq (Y_1, Y_2, \ldots, Y_L)$ and $\Theta \triangleq (\Theta_1, \Theta_2, \ldots, \Theta_L)$. For the sake of definiteness, we assume that the channel input symbols X_k are M-PSK symbols (cf. Fig. 2.7) and are protected by a binary low-density parity check (LDPC) code.[1] The coded channel input symbols X_k are transmitted in

[1]We refer to Appendix C for more information about LDPC-codes.

frames of L symbols. We consider the two phase models we described in Chapter 2, i.e., the constant-phase model and the random-walk phase model.

Constant Phase: $\Theta_k = \Theta_0 \in [0, 2\pi)$, an unknown constant.

Random Walk:

$$\Theta_k = (\Theta_{k-1} + W_k) \bmod 2\pi, \tag{4.2}$$

where W_k is white Gaussian noise with known variance σ_W^2.

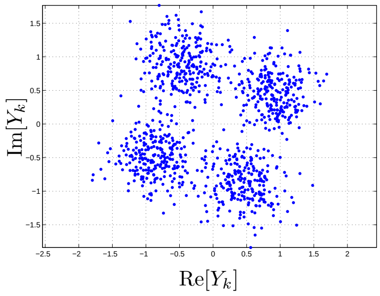


Figure 4.1: Realization of Y in the constant-phase model; the figure depicts the symbols Y_k in the complex plane ($\sigma_N = 0.3$, $\theta = 0.3$, $L = 1000$ and $M = 4$).

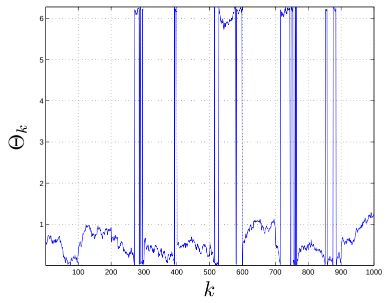
Fig. 4.1 depicts a realization of Y in the constant-phase model. A realization of Θ and Y in the random-walk phase model is shown in Fig. 4.2(a) and Fig. 4.2(b) respectively.

The algorithms we propose are approximations of the symbol-wise MAP (maximum a posteriori) decoder (cf. Appendix A):

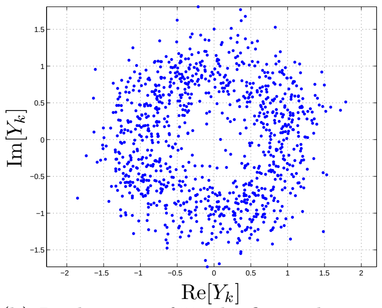
$$\hat{X}_k^{\text{MAP}} = \underset{x_k}{\operatorname{argmax}} \int_0^{2\pi} \ldots \int_0^{2\pi} p(x_k, y, \theta)\, d\theta \tag{4.3}$$

$$= \underset{x_k}{\operatorname{argmax}} \int_0^{2\pi} \ldots \int_0^{2\pi} \sum_{x \text{ with } x_k \text{ fixed}} p(x, y, \theta)\, d\theta, \tag{4.4}$$

The function $p(x, y, \theta)$ stands for the joint probability function of X, Y, and Θ; it is a probability density function (pdf) in θ and y and a



(a) Realization of Θ.



(b) Realization of Y; the figure depicts the symbols Y_k in the complex plane.

Figure 4.2: Random-walk phase model ($\sigma_N = 0.3$, $\theta_1 = 0.5$, $L = 1000$, $M = 4$, and $\sigma_W = 0.05$).

probability mass function (pmf) in x. Note that Equations (4.3) and (4.4) involve averaging over the phase Θ.

As outlined already by Wiberg in '96 [210], iterative (message-passing) algorithms for joint decoding and channel estimation may be derived from the factor graph of the code and the channel (see also [214]). We will follow this principle to derive algorithms for carrier-phase estimation. Message-passing algorithms to approximately compute (4.3) and (4.4) may be obtained by the following procedure:

a) The probability function $p(x, y, \theta)$ is represented by a factor graph.

b) Message types are chosen and message-update rules are computed.

c) A message-update schedule is chosen.

In Step 2, finite-alphabet variables (such as X_k) are handled by the standard sum-product rule. For continuous-valued variables (such as Θ_k), however, the sum-product rule leads to intractable integrals, which can be approximated in several ways; each such approximation corresponds to a certain message type and results in a different phase-estimation algorithm. In some cases, we will obtain an approximation of the entire posterior distribution $p(\theta|\, y)$; in other cases, we will obtain only an estimate $\hat{\theta}$.

Phase estimation is of course an old subject[2] and several algorithms for joint iterative decoding and phase estimation have recently appeared, both for the constant-phase model as for the random-walk phase model.

- Constant-phase model:
  A large number of synchronization algorithms for the constant-phase model have been proposed. Some of the algorithms are ad-hoc [28] [35] [27] [189], others are based on the expectation maximization algorithm [141] [143] [185] [144] [185] [122] [82] or the sum-product algorithm [152] [153].
- Random-walk phase model:
  In parallel work, Colavolpe et al. derived other phase estimators from the factor graph of the random-walk model [36]. They approximated the sum-product messages by so-called "canonical distributions" such as the Gaussian and Tikhonov distribution. In parallel work, Noels et al. [151] derived EM-based algorithms for the random-walk model. A turbo-synchronization algorithm for an other simple stochastic phase model was presented in [137].

This chapter is structured as follows. In Section 4.2, we explain the factor graphs. We derive the sum-product message-update rules in Section 4.3. We elaborate on the update schedule in Section 4.4. The various message-passing algorithms are described in Section 4.5 through Section 4.9; in each of those sections, we first describe the considered estimation method as message passing on factor graphs, then we apply the method to the carrier-phase estimation problem. Simulation results are presented in Section 4.10. A summary of this chapter is given in Section 4.11.

4.2 Factor Graphs of the System

The system described in Section 8.1 is easily translated into the factor graph of Fig. 4.3, which represents the factorization of the joint probability function of all variables in the system. The upper part of the graph is the indicator function of the LDPC code, with parity check nodes in the top row that are "randomly" connected to equality constraint nodes ("bit nodes"). The function f is the deterministic mapping

[2]See, e.g., [197] for one of the first studies on this subject.

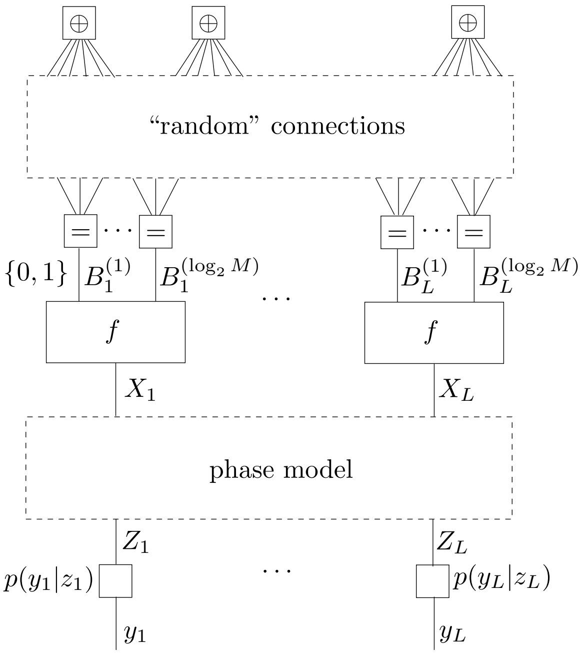


Figure 4.3: Factor graph of LDPC code and the channel model.

$f : \left(B_k^{(1)}, \ldots, B_k^{(\log_2 M)}\right) \mapsto X_k$ of the (encoded) bits $B_k^{(1)}, \ldots, B_k^{(\log_2 M)}$ to the (channel) symbol X_k. The nodes labeled "f" ("bit mapper nodes") correspond to the factors

$$\delta_f\left(b_k^{(1)}, \ldots, b_k^{(\log_2 M)}, x_k\right) \tag{4.5}$$

$$\triangleq \begin{cases} 1, & \text{if } f\left(b_k^{(1)}, \ldots, b_k^{(\log_2 M)}\right) = x_k; \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

The bottom row of the graph represents the factors

$$p(y_k|z_k) \triangleq (2\pi\sigma_N^2)^{-1}\, e^{-|y_k - z_k|^2/2\sigma_N^2}. \tag{4.7}$$

The "phase model" in Figure 4.3 is detailed in Figures 4.4 and 4.5. In these figures, S_k is defined as $S_k \triangleq e^{j\Theta_k}$ and Z_k is defined as $Z_k \triangleq X_k S_k$. The top row of nodes ("multiply nodes") in Figures 4.4 and 4.5

represents the factors $\delta(z_k - x_k s_k)$. The function g is the deterministic mapping $g : \Theta_k \mapsto S_k$ of the phase Θ_k to S_k; the nodes labeled "g" in Figures 4.4 and 4.5 represent the factors $\delta(s_k - e^{j\theta_k})$. The compound equality constraint node in Figure 4.4 imposes the constraint $\Theta_k = \Theta$, $\forall k$ (cf. Remark 3.2). In Fig. 4.5, the nodes labeled $p(\theta_k|\theta_{k-1})$ ("phase noise nodes") represent the factors

$$p(\theta_k|\theta_{k-1}) \triangleq (2\pi\sigma_W^2)^{-1/2} \sum_{n\in\mathbf{Z}} e^{-((\theta_k-\theta_{k-1})+n2\pi)^2/2\sigma_W^2}. \tag{4.8}$$

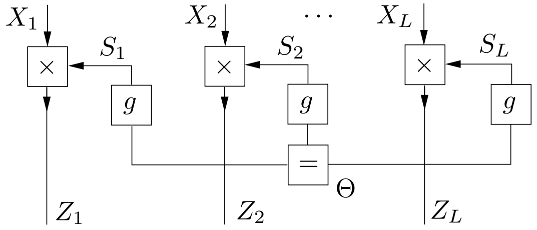


Figure 4.4: Factor graph of the constant-phase model.

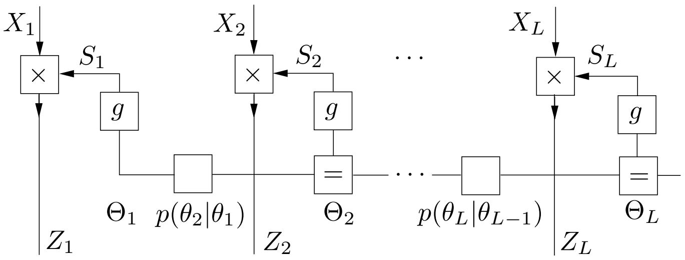


Figure 4.5: Factor graph of the random-walk phase model.

4.3 Sum-Product Message Update Rules

We now apply the sum-product algorithm to the factor graph in Fig. 4.3, where the factor graph of the phase model is detailed in Fig. 4.4 and Fig. 4.5.

In this section, we compute the update rules; in Section 4.4, we explain in which order the messages are updated.

The messages out of the nodes $p(y_k|z_k)$ are the functions $p(y_k|z_k)$ themselves. The computation of the messages out of the bit mapper nodes and inside the graph of the LDPC code is standard [119]; we therefore only consider the computation of the messages inside, and out of, the graph of the two phase models.

Straightforward application of the sum-product algorithm to the graph of the two phase models results in the following update rules :

- **Equality Constraint Node** (see Fig. 4.6(a))

$$\begin{aligned}\mu_{\boxed{=}\to\Theta}(\theta) &\propto \int_{\theta'}\int_{\theta''}\delta(\theta-\theta')\delta(\theta-\theta'') \\ &\qquad \cdot\mu_{\Theta'\to\boxed{=}}(\theta')\mu_{\Theta''\to\boxed{=}}(\theta'')d\theta' d\theta'' \qquad (4.9)\\ &= \mu_{\Theta'\to\boxed{=}}(\theta)\mu_{\Theta''\to\boxed{=}}(\theta), \qquad (4.10)\end{aligned}$$

  where Θ, Θ', and $\Theta'' \in [0, 2\pi)$. Note that the message $\mu_{\boxed{=}\to\Theta}(\theta)$ is defined up to some scaling factor (cf. Remark 3.3). The messages along the edges Θ' and Θ'' are computed analogously.

- **Multiply Node** (see Fig. 4.6(b))

$$\begin{aligned}\mu_{\boxtimes\to S}(s) &\propto \sum_X\int_z \mu_{X\to\boxtimes}(x)\mu_{Z\to\boxtimes}(z)\delta(z-xs)dz \qquad (4.11)\\ &= \sum_X \mu_{X\to\boxtimes}(x)\mu_{Z\to\boxtimes}(xs), \qquad (4.12)\\ \mu_{\boxtimes\to X}(x) &\propto \int_z \oint_{\text{unit circle}} \mu_{S\to\boxtimes}(s)\mu_{Z\to\boxtimes}(z)\delta(z-xs)dsdz \qquad (4.13)\\ &= \oint_{\text{unit circle}} \mu_{S\to\boxtimes}(s)\mu_{Z\to\boxtimes}(xs)ds, \qquad (4.14)\end{aligned}$$

  where X is an M-PSK symbol, $Z \in \mathbb{C}$, S takes values on the unit circle, and the line integral in the RHS of (4.14) is computed over the unit circle.

- **Phase Noise Node** (see Fig. 4.6(c))

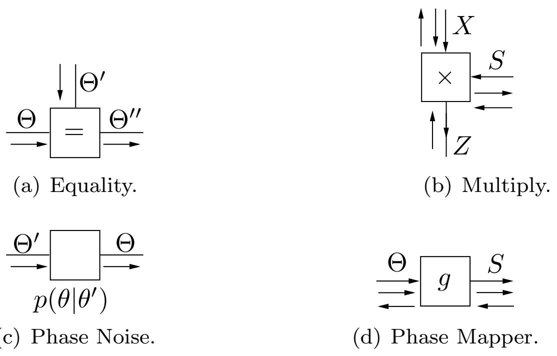
$$\begin{aligned}\mu_{p\to\Theta}(\theta) &\propto \int_0^{2\pi} \mu_{\Theta'\to p}(\theta')p(\theta|\theta')d\theta', &(4.15)\\ &= \int_0^{2\pi} \mu_{\Theta'\to p}(\theta')\sum_{n\in\mathbf{Z}} e^{-((\theta-\theta')+n2\pi)^2/2\sigma_W^2}d\theta', &(4.16)\end{aligned}$$

where Θ, and $\Theta' \in [0, 2\pi)$. The message $\mu_{p\to\Theta'}(\theta')$ along the edge Θ' is computed analogously.

- **Phase Mapper Node** (see Fig. 4.6(d))

$$\begin{aligned}\mu_{g\to S}(s) &= \mu_{\Theta\to g}(\arg s), &(4.17)\\ \mu_{g\to\Theta}(\theta) &= \mu_{S\to g}(e^{j\theta}), &(4.18)\end{aligned}$$

where $\arg s$ stands for the argument of s, $\Theta \in [0, 2\pi)$ and S takes values on the unit circle.



(a) Equality. (b) Multiply. (c) Phase Noise. (d) Phase Mapper.

Figure 4.6: Nodes in the graph of the phase models.

Since the messages $\mu_{\Theta_k\to g}$ and $\mu_{S_k\to\boxtimes}$ in Figures 4.4 and 4.5 encode the same information, there is no need to compute (and store) the messages $\mu_{S_k\to\boxtimes}$ explicitly. The messages $\mu_{\boxtimes\to X_k}$ can *directly* be computed from $\mu_{\Theta_k\to g}$ rather than from $\mu_{S_k\to\boxtimes}$, as illustrated in Fig. 4.7. The corresponding update rule is obtained by clustering the multiply and phase mapper nodes (cf. Section 3.3). The standard sum-product rule

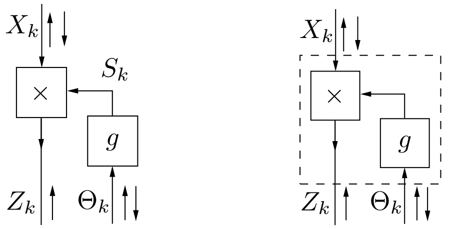


Figure 4.7: Multiply and mapper node. (left) the two nodes are treated separately, and the messages along the edges S_k are computed explicitly; (right) the two nodes are clustered.

is applied to the resulting (compound) node (dashed box in Fig. 4.7 (right)):

$$\begin{aligned}
\mu_{\boxtimes \to X_k}(x_k) &\propto \int_0^{2\pi} \int_{\text{unit circle}} \int_{z_k} \delta(z_k - x_k s_k)\delta\big(s_k - e^{j\theta_k}\big) \\
&\qquad \mu_{\Theta_k \to g}(\theta_k)\mu_{Z_k \to \boxtimes}(z_k)\, d\theta_k\, ds_k\, dz_k, &(4.19)\\
&= \int_0^{2\pi} \mu_{\Theta_k \to g}(\theta_k)\mu_{Z_k \to \boxtimes}\big(x_k e^{j\theta_k}\big)\, d\theta_k, &(4.20)\\
&\propto \int_0^{2\pi} \mu_{\Theta_k \to g}(\theta_k) e^{-|x_k e^{j\theta_k} - y_k|^2/2\sigma_N^2}\, d\theta_k, &(4.21)
\end{aligned}$$

which is more practical to implement than (4.14). Similarly, one can rewrite the update rule for the messages $\mu_{g\to\Theta_k}(\theta_k)$ out of the phase mapper node along the Θ_k edges as

$$\begin{aligned}
\mu_{g\to\Theta_k}(\theta_k) &\propto \sum_{x_k} \mu_{X_k \to \boxtimes}(x_k)\mu_{Z_k \to \boxtimes}\big(x_k e^{j\theta_k}\big) &(4.22)\\
&\propto \sum_{x_k} \mu_{X_k \to \boxtimes}(x_k) e^{-|x_k e^{j\theta_k} - y_k|^2/2\sigma_N^2}. &(4.23)
\end{aligned}$$

According to the sum-product rule, the messages $\mu_{\Theta_k \to g}(\theta_k)$ in Fig. 4.4 are computed as

$$\mu_{\Theta_k \to g}(\theta_k) \propto \prod_{\ell \neq k} \mu_{g \to \Theta_\ell}(\theta_k), \qquad (4.24)$$

i.e., the product of all messages arriving at the equality constraint node, except the one that arrives along the edge Θ_k. However, it is more convenient to approximate the messages $\mu_{\Theta_k \to g}(\theta_k)$ by the product of *all* messages arriving at the equality constraint node:

$$\begin{aligned} \mu_{\Theta_k \to g}(\theta_k) &\approx \prod_{i=1}^{L} \mu_{g \to \Theta_i}(\theta_k) & (4.25) \\ &\propto \mu_{\boxed{=} \to \Theta}(\theta_k), & (4.26) \end{aligned}$$

where $\mu_{\boxed{=} \to \Theta}(\theta_k)$ is the message leaving the equality node along the Θ edge. The approximation (4.25)–(4.26) is in practice satisfactory, since L, the number of messages in the product (4.24) and (4.25), is typically large (between 100 and 10.000).

The integrals in the RHS of (4.15) and (4.21) are *intractable*. In the following sections, we will describe several approximative integration methods:

- numerical integration (Section 4.5),
- particle methods (Section 4.6),
- adaptive quantization (Section 4.7),
- gradient methods (Section 4.8),
- expectation maximization (Section 4.9).

Each approximation leads to a different message-passing algorithm for code-aided phase estimation. Before we describe each algorithm individually, we explain in which order the messages in Fig. 4.3 are updated, since this order is common to all phase estimators. We treat the scheduling inside the phase model (see Fig. 4.4 and Fig. 4.5) when we describe the individual phase estimators.

4.4 Scheduling

The messages in the graph of Fig. 4.8 are updated in the following order:

① The messages from the nodes $p(y_k|z_k)$ towards the phase model.

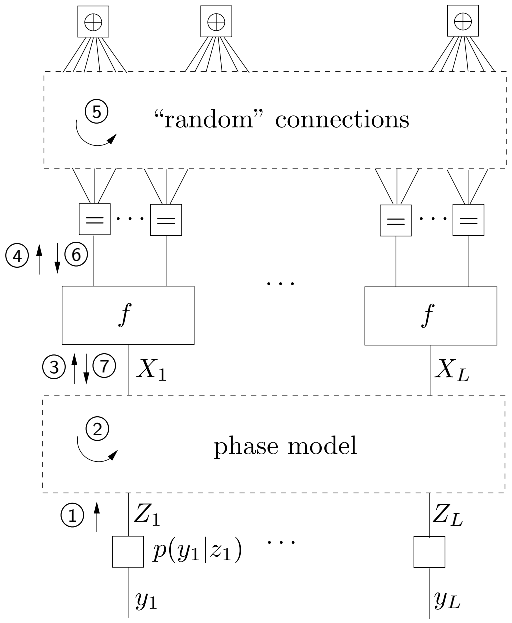


Figure 4.8: Message update schedule. The circled numbers indicate the order of the message computations.

② One or more iterations in the graph of the phase model (Fig. 4.4 or Fig. 4.5).

③ The messages from the phase model towards the mapper nodes f.

④ The messages from the mapper nodes f towards the bit nodes.

⑤ One or more iterations of the LDPC decoder.

⑥ The messages from the bit nodes towards the mapper nodes f.

⑦ The messages from the mapper nodes f towards the phase model.

The updates ②–⑦ are iterated until convergence or until the available time is over.

4.5 Numerical Integration

4.5.1 General Idea

Numerical integration (or "grid based integration" or "quantization") is a well-known technique to evaluate intractable integrals. Continuous variables X are (uniformly) quantized, and the (intractable) integral-product rule is replaced by a finite sum. If we apply the simplest numerical integration scheme, i.e., the rectangular rule, the integral-product rule is evaluated as follows.

Integral-product rule evaluated by numerical integration:

$$\begin{aligned}\mu_{f\to Y}(y) \quad &\overset{\triangle}{\propto} \quad \sum_{i_1,\ldots,i_N} f\big(y, \hat{x}_1^{(i_1)}, \ldots, \hat{x}_N^{(i_N)}\big) \cdot \\ &\qquad\qquad \mu_{X_1\to f}\big(\hat{x}_1^{(i_1)}\big)\cdots\mu_{X_N\to f}\big(\hat{x}_N^{(i_N)}\big), \end{aligned} \tag{4.27}$$

where $\hat{x}_k^{(i_k)}$ is the i_k-th quantization level of X_k.

The integrand in the integral-product rule, i.e.,

$$g(y, x_1, \ldots, x_N) \overset{\triangle}{=} f(y, x_1, \ldots, x_N) \cdot \mu_{x_1\to f}(x_1)\cdots\mu_{x_N\to f}(x_N) \tag{4.28}$$

is approximated as a piecewise constant function (w.r.t. $x_1, \ldots, x_N$), as illustrated in Fig. 4.9. The messages $\mu_{X_k\to f}$ are represented by their

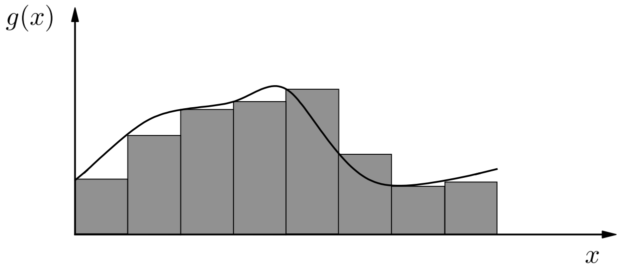


Figure 4.9: Rectangular integration rule.

function values at the quantization levels $\hat{x}_k^{(i_k)}$ (see Fig. 4.10); we refer to this message representation as **quantized message**.

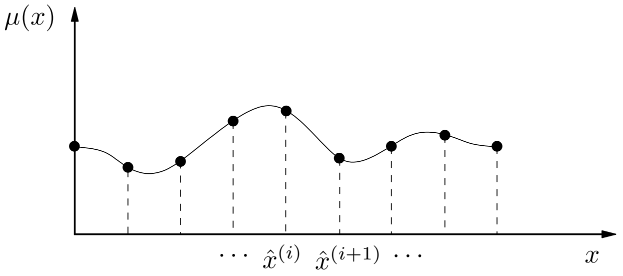


Figure 4.10: Quantized message.

Remark 4.1. (Higher-order integration rules)
In (4.27) we used the rectangular integration rule, which approximates the integrand by piecewise constant functions. One may apply more accurate integration schemes based on higher-order polynomials [164] or splines [196] . We verified experimentally that the rectangular rule (4.27) suffices for our purposes (i.e., phase estimation): the performance gain due to higher-order integration rules turns out to be negligible. This is not necessarily the case for other estimation problems.

4.5.2 Application

Constant-Phase Model

The forward messages μ_k^F and downward messages $\mu_{g\to\Theta_k}$ are quantized messages (see Fig. 4.11). The messages $\mu_{\Theta_k\to g}(\theta_k)$ are approximated by the message $\mu_{\boxminus\to\Theta}(\theta) \stackrel{\triangle}{=} \mu_L^F$ (cf. (4.25)–(4.26)). The latter is computed in a *forward* sweep, as illustrated in Fig. 4.11, as a consequence, the backward messages $\mu_{g\to\Theta_k}$ are not required.

The update schedule is as follows:

① The message μ_1^F is initialized (cf. (4.23)):

$$\mu_1^F(\theta^{(i)}) = \mu_{g\to\Theta_1}(\theta^{(i)}), \tag{4.29}$$

where the quantization levels $\theta^{(i)}$ are defined as $\theta^{(i)} \triangleq 2\pi i/N$, and $i = 1, \ldots, N$.

② The messages μ_k^F are updated in a forward sweep:

$$\mu_k^F(\theta^{(i)}) \propto \mu_{k-1}^F(\theta^{(i)})\mu_{g\to\Theta_k}(\theta^{(i)}), \tag{4.30}$$

for $i = 1, \ldots, N$ and $k = 2, \ldots, L$.

③ The upward messages $\mu_{\boxtimes\to X_k}(x_k)$ are computed from the message μ_L^F (cf. (4.21)):

$$\mu_{\boxtimes\to X_k}(x_k) \overset{\triangle}{\propto} \sum_{i=1}^{N} \mu_L^F(\theta^{(i)})\mu_{Z_k\to\boxtimes}\big(x_k \exp(j\hat{\theta}^{(i)})\big). \tag{4.31}$$

for $k = 1, \ldots, L$.

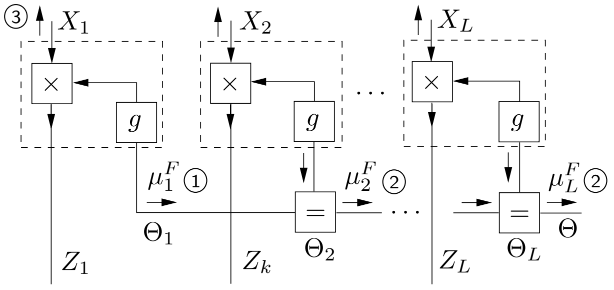


Figure 4.11: Quantized messages in the factor graph of the constant-phase model.

Random-Walk Phase Model

All messages along the Θ_k edges (see Fig. 4.12) are quantized messages, i.e., $\mu_{p_k\to\Theta_{k-1}}$, $\mu_{p_k\to\Theta_k}$, $\mu_{\Theta_k\to g}$ and $\mu_{g\to\Theta_k}$. The updates are scheduled as follows:

① The messages $\mu_{p_k\to\Theta_k}$ and $\mu_{p_k\to\Theta_{k-1}}$ are updated in a forward and backward sweep respectively. At the phase noise nodes, one computes the forward messages as follows:

$$\mu_{p_k \to \Theta_k}(\hat{\theta}^{(i)}) \quad \overset{\triangle}{\propto} \quad \sum_{j=1}^{N} \mu_{\Theta_{k-1} \to p_k}(\hat{\theta}^{(j)}) p_k(\hat{\theta}^{(i)}|\hat{\theta}^{(j)}), \quad (4.32)$$

for all $i = 1, \ldots, N$. The backward messages $\mu_{p_k \to \Theta_{k-1}}$ are computed similarly. At the equality constraint nodes, the messages are updated as in (4.30).

② The upward messages $\mu_{\Theta_k \to g}$ are computed:

$$\mu_{\Theta_k \to g}(\theta^{(i)}) \propto \mu_{p_{k-1} \to \Theta_k}(\theta^{(i)}) \mu_{p_k \to \Theta_k}(\theta^{(i)}), \quad (4.33)$$

③ The messages $\mu_{\boxtimes \to X_k}(x_k)$ are obtained from the messages $\mu_{\Theta_k \to g}$:

$$\mu_{\boxtimes \to X_k}(x_k) \overset{\triangle}{\propto} \sum_{i=1}^{N} \mu_{\Theta_k \to g}(\theta^{(i)}) \mu_{Z_k \to \boxtimes}\big(x_k \exp(j\hat{\theta}^{(i)})\big). \quad (4.34)$$

for $k = 1, \ldots, L$.

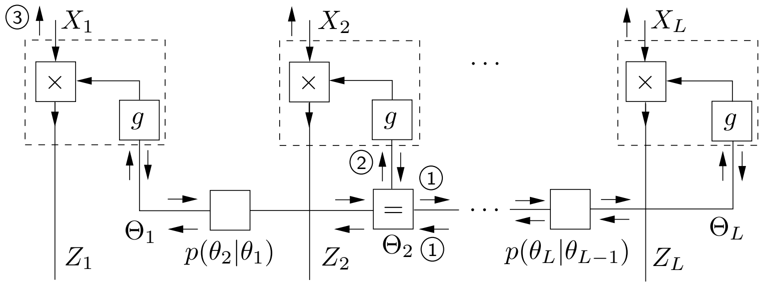


Figure 4.12: Quantized messages in the factor graph of the random-walk phase model.

4.5.3 Summary

In the numerical-integration approach:

- Continuous variables are (usually uniformly) **quantized**.

- Integrals are replaced by **finite sums**.

- Messages are represented by their function values evaluated at the quantization levels (**"quantized messages"**).

4.6 Particle Methods

4.6.1 General Idea

A probability (density or mass) function f can be represented by a list of samples ("particles") from f, as illustrated in Fig. 4.13.[3] This data type

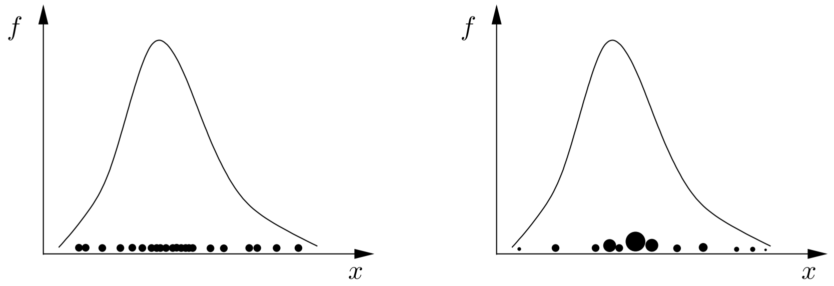


Figure 4.13: A probability density function f and its representation as a list of particles. The radius of the particles is proportional to their weights. (left) uniform weights; (right) non-uniform weights.

is the foundation of Monte-Carlo methods [171] (or "particle methods"). Integrals are evaluated as (weighted) averages over lists of samples.

[3]It is sometimes useful to approximate a probability *mass* function by a list of samples, especially, if the alphabet of the discrete random variable is large.

Integral-product rule evaluated by particle methods:

$$\mu_{f\to Y}(y) \overset{\triangle}{\propto} \sum_{i_1,\ldots,i_N} f(y, \hat{x}_1^{(i_1)}, \ldots, \hat{x}_N^{(i_N)}) \cdot w_1^{(i_1)} \cdots w_N^{(i_N)}, \quad (4.35)$$

where $\hat{x}_k^{(i_k)}$ is the i_k-th particle of the particle list that represents $\mu_{X_k\to f}$, and $w_k^{(i_k)}$ is the weight of that particle.

In the update rule (4.35), samples from the sum(integral)-product messages $\mu_{X_k\to f}$ are required. We now investigate how such samples can be obtained. Suppose we wish to draw samples from the sum(integral)-product message $\mu_{g\to X}$ out of the generic node g (see Fig. 4.14):

$$\mu_{g\to X}(x) \overset{\triangle}{=} \sum_{z_1,\ldots,z_N} g(x, z_1, \ldots, z_N)\mu_{Z_1\to g}(z_1)\cdots\mu_{Z_N\to g}(z_N), \quad (4.36)$$

where $\sum$ stands for summation if Z_k is discrete and for integration otherwise. First we consider the case where the variables Z_k are discrete, then

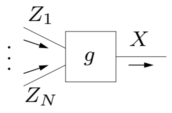


Figure 4.14: Message along a generic edge.

we investigate the continuous case.

Discrete variables Z_k

One may generate samples from $\mu_{g\to X}$ by the following procedure:

a) For each k, draw a value $\hat{z}_k$ of Z_k with probability proportional to $\mu_{Z_k\to g}(\hat{z}_k)$.

b) Draw a sample $\hat{x}$ from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$.

c) Iterate 1–2 until a sufficient number of samples are obtained.

Note that the resulting samples have uniform weights. We therefore refer to the above sampling method (Step 1–3) as **unweighted sampling**.

Alternatively, one may draw samples $\hat{x}$ from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$ for each valid configuration $(\hat{z}_1, \ldots, \hat{z}_N)$. The weight w of sample $\hat{x}$ is proportional to

$$w \propto \prod_{k=1}^{N} \mu_{Z_k \to g}(\hat{z}_k). \tag{4.37}$$

This sampling method is called **weighted sampling**. From the resulting list of weighted samples, one can generate a list of *uniform* samples by a technique called **resampling**: one draws samples $\hat{x}$ from the weighted list with probability proportional to the weights w; a particle with large weight w may be drawn several times, whereas a particle with small weight may not be drawn at all. Note that weighted sampling followed by resampling leads to unweighted samples; hence, weighted sampling in combination with resampling is an alternative to unweighted sampling.

Remark 4.2. (Drawing samples)
In weighted as well as unweighted sampling, we need to draw samples from a function $g(x, \hat{z}_1, \ldots, \hat{z}_N)$. Note that an *explicit* form of g is not required. Samples $\hat{x}$ can be obtained by *simulating* the node operation g ("direct sampling"); in certain systems, e.g., discrete-time state-space models, this may correspond to the integration of a stochastic differential equation. If it is hard to sample from g *directly*, we may apply *importance sampling* or *Markov-Chain Monte-Carlo methods* (MCMC). We will describe those two sampling techniques in Section 4.6.3 and 4.6.5 respectively.

Continuous variables Z_k

If the variables Z_k are **continuous**, we distinguish the following cases:

- If a **closed-form expression** of the integral (4.36) is available, one may sample from $\mu_{g \to X}$ by standard techniques (e.g., importance sampling or MCMC).
- If $\mu_{Z_k \to g}$ are **quantized messages**, one proceeds as in the case of discrete variables Z_k.
- If the messages $\mu_{Z_k \to g}$ are **lists of samples**, the procedure is also very similar to the one in the discrete case. The first step in the unweighted sampling procedure is slightly modified: for each k, one

draws a particle $\hat{z}_k$ with probability proportional to its weight w_k. In weighted sampling, one draws a sample from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$, for each N-tuple of particles $(\hat{z}_1, \ldots, \hat{z}_N)$. The weight w of this sample is proportional to

$$w \propto \prod_{k=1}^{N} w_k, \tag{4.38}$$

where w_k is the weight of particle $\hat{z}_k$.

- If the incoming messages are **single values** $\hat{z}_k$, one draws samples from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$.
- **Combinations** of the previous cases are possible.

Specific node functions

So far, we have considered generic node functions g. We now apply the above generic rules to two important classes of node functions: deterministic mappings and equality constraint nodes.

- **(Deterministic mapping)**
  Suppose that the function g corresponds to a deterministic mapping, for example

  $$g(x, z_1, \ldots, z_N) \triangleq \delta(x - h(z_1, \ldots, z_N)), \tag{4.39}$$

  where h maps the N-tuple $(z_1, \ldots, z_N)$ to x. Samples $\hat{x}$ from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$ are all *identical*, i.e., $\hat{x} = h(\hat{z}_1, \ldots, \hat{z}_N)$.

- **(Equality constraint node)**
  Suppose that the node g is an equality constraint node, i.e.,

  $$g(x, z_1, \ldots, z_N) \triangleq \delta(x - z_1) \prod_{k=1}^{N-1} \delta(z_{k+1} - z_k). \tag{4.40}$$

  The outgoing message $\mu_{g\to X}(x)$ is given by:

  $$\mu_{g\to X}(x) \propto \prod_{k=1}^{N} \mu_{Z_k\to g}(x). \tag{4.41}$$

- If X is **discrete**, one draws a sample $\hat{x}$ with probability proportional to $\prod_{k=1}^{N} \mu_{Z_k \to g}(\hat{x})$.
- If X is continuous, and **closed-form expressions** for the incoming messages $\mu_{Z_k \to g}$ are available, one may sample from the product (4.41). This can be done in an elegant fashion by importance sampling, as we will see later on.
- If the messages $\mu_{Z_k \to g}$ are **quantized messages**, one proceeds as in the discrete case.
- If the messages $\mu_{Z_k \to g}$ are represented as **lists of samples**, it is not straightforward to draw samples from (4.41). One usually first generates a continuous representation such a density trees or a mixture of Gaussian distributions for each of the incoming messages $\mu_{Z_k \to g}$. Efficient methods have been devised to draw samples from products of such density approximations [88] . However, those methods are rather complicated, and it is therefore recommendable to *avoid* products of particle lists. This is in practice often possible, as we illustrate in Section 4.6.4 (cf. Remark 4.5).
- Also **combinations** of the previous situations are possible, as we will illustrate in Section 4.6.3 by the example of importance sampling.
- If the incoming messages are **single values** $\hat{z}_k$, there is no reasonable way to draw samples from (4.41). However, one may compute the outgoing message $\mu_{g \to X}$ as the average of the incoming estimates $\hat{z}_k$:

$$\mu_{g \to X} \triangleq \frac{1}{N} \sum_{k=1}^{N} \hat{z}_k. \tag{4.42}$$

In the following, we describe five standard Monte-Carlo techniques as instances of the above generic rules, i.e.,

- Gibbs sampling,
- importance sampling,
- particle filtering (“sequential Monte-Carlo filtering”),
- Markov-Chain Monte-Carlo methods,
- simulated annealing.

4.6.2 Gibbs Sampling

Suppose we wish to draw samples from a multivariate probability function $f(x_1, x_2, \ldots, x_N)$. This can be done by the following iterative algorithm known as Gibbs sampling [171, p. 371–407]:

a) Choose an initial value $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$.

b) Choose an index k.

c) Draw a sample $\hat{x}_k$ from

$$f(x_k) \triangleq \frac{f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)}{\sum_{x_k} f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)}. \tag{4.43}$$

d) Iterate 2–3 a "large" number of times.

The values $\hat{x}$ from the first n iterations are typically discarded (e.g., n=100), since they depend more on the initial value $\hat{x}$ than on the target distribution f.

One may obtain all samples in a single run, i.e., one continues iterating after the n initial iterations until a sufficient number of samples are obtained. Alternatively, each sample (or a fraction of the samples) may be generated in a separate run.

If the function $f(x_1, x_2, \ldots, x_N)$ has "structure", it may become easier to generate samples $\hat{x}_k$ (cf. Step 3). Let us have a look at a warming-up example.

Example 4.1. (Gibbs sampling as message passing)
Suppose the global function $f(x, y)$ factorizes as

$$f(x, y) \triangleq f_A(x, y) f_B(y), \tag{4.44}$$

as depicted in Fig. 4.15(a).

Gibbs sampling from (4.44) boils down to the iterative computation of the following messages on the factor graph of Fig. 4.15(a):

a) Message $\boldsymbol{\mu_X(x)}$ **arriving** at X edge:

$$\mu_X(x) \triangleq f_A(x, \hat{y}) \tag{4.45}$$

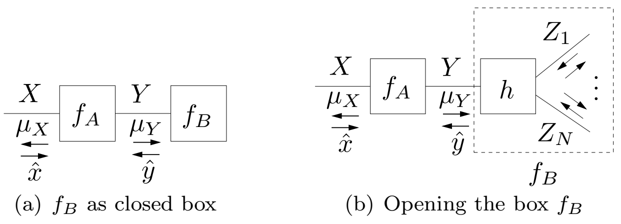


Figure 4.15: Gibbs sampling as message passing.

b) Message $\boldsymbol{\hat{x}}$ **broadcast** by X:
Sample $\hat{x}$ from:

$$f(x) \triangleq \frac{\mu_X(x)}{\sum_x \mu_X(x)}. \tag{4.46}$$

c) Message $\boldsymbol{\mu_Y(y)}$ **arriving** at Y edge:

$$\mu_Y(y) \triangleq f_A(\hat{x}, y) \tag{4.47}$$

d) Message $\boldsymbol{\hat{y}}$ **broadcast** by Y:
Sample $\hat{y}$ from:

$$f(y) \triangleq \frac{f_B(y)\mu_Y(y)}{\sum_y f_B(y)\mu_Y(y)}. \tag{4.48}$$

The node f_B may in its turn have internal structure, as depicted in Fig. 4.15(b). The message $f_B(y)$ is then a "summary" of the graph behind it, computed by applying the generic sum-product rule (or some approximation) to the incident node h. In particular, if Gibbs sampling is applied in the "box" f_B (dashed box in Fig. 4.15(b)), the message $f_B(y)$ is given by:

$$f_B(y) = h(y, \hat{z}_1, \ldots, \hat{z}_N), \tag{4.49}$$

where $\hat{z}_k$ $(k = 1, \ldots, N)$ is a sample from $f(z_k)$ defined as:

$$f(z_k) \triangleq \frac{\mu_{h\to Z_k}(z_k)\mu_{Z_k\to h}(z_k)}{\sum_{z_k} \mu_{h\to Z_k}(z_k)\mu_{Z_k\to h}(z_k)}. \tag{4.50}$$

$\square$

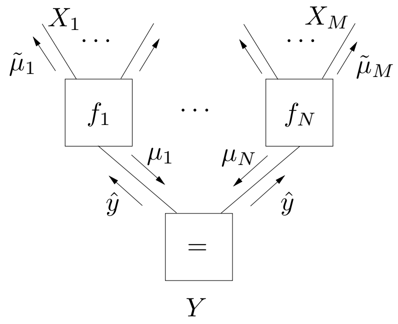


Figure 4.16: Gibbs sampling at a generic edge Y.

The previous example can easily be extended to general factorizations f. Gibbs sampling can be interpreted as a message-passing algorithm that operates on a factor graph of f by iterating the following steps:

a) Select a variable (equality constraint node) Y in the factor graph of f (see Fig. 4.16).

b) The equality constraint node Y generates the message $\hat{y}$ by sampling from:
$$f(y) \triangleq \frac{\mu_1(y) \ldots \cdot \mu_N(y)}{\sum_y \mu_1(y) \ldots \cdot \mu_N(y)}, \tag{4.51}$$
and broadcasts $\hat{y}$ to its neighboring nodes f_k $(k = 1, \ldots, M)$.

c) The nodes f_k update their outgoing messages $\tilde{\mu}$ by applying the sum-product rule with as incoming messages the samples $\hat{y}$ and $\hat{x}_\ell$ $(\ell = 1, \ldots, M)$ (cf. (4.84)).

The message-passing algorithm of Example 4.1 is an instance of the above generic scheme, as easily can be verified. In conclusion: Gibbs sampling is equivalent to message passing on a factor graph, where *each* message is represented by a single sample; obviously, one may apply Gibbs sampling *locally* in a graph (and apply other approximation techniques in the other parts of the graph).

Remark 4.3. (Gibbs sampling from joint densities)
In order to speed up the Gibbs sampling algorithm, one may *group* several variables, i.e., one may sample several variables *jointly* [93, p. 201].

For instance, one could sample x_k and x_{k+1} from the joint probability function

$$f(x_k, x_{k+1}) \triangleq \frac{f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, x_{k+1}, \hat{x}_{k+2}, \ldots, \hat{x}_N)}{\sum_{x_k} f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, x_{k+1}, \hat{x}_{k+2}, \ldots, \hat{x}_N)}. \tag{4.52}$$

Note that this idea is only interesting if x_k and x_{k+1} are *dependent* conditioned on $(\hat{x}_1, \ldots, \hat{x}_{k-1}, \hat{x}_{k+2}, \ldots, \hat{x}_N)$, i.e., if the edges x_k and x_{k+1} are incident to a common node. Indeed, if x_k and x_{k+1} are conditionally *independent* (or equivalently, the corresponding edges are not incident to the same node), the function $f(x_k, x_{k+1})$ factorizes as

$$f(x_k, x_{k+1}) \triangleq f(x_k) f(x_{k+1}), \tag{4.53}$$

and sampling from $f(x_k, x_{k+1})$ reduces to *independent* sampling from the marginals $f(x_k)$ and $f(x_{k+1})$. Grouping can naturally be expressed graphically (see Fig. 4.17(a)). The pair $(\hat{y}, \hat{z})$ is drawn from the (normalized) product of the function $h(y, z)$ (dashed box) and the messages arriving at the box h along the edges Y and Z. Note that the box h may be a summary of a subgraph, as suggested in Fig. 4.17(a); the function $h(y, z)$ is obtained by summarizing over the internal variables of the box h. Sampling Y and Z jointly is equivalent to "standard" sampling (cf. Step 2) of the clustered variable (Y, Z) (see Fig. 4.17(b)).

The extension of (4.53) to higher-order joint densities is straightforward. As the order of the joint density increases, however, it often becomes more difficult to sample from this density.

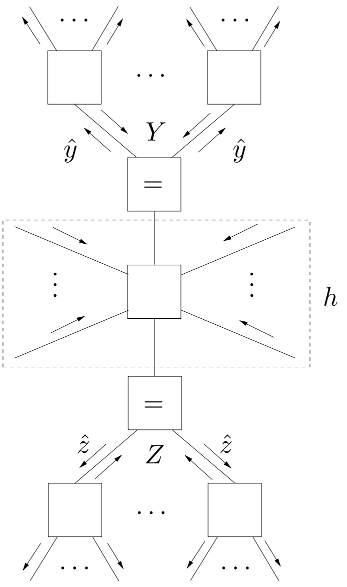
4.6.3 Importance Sampling

Suppose we wish to compute the expectation

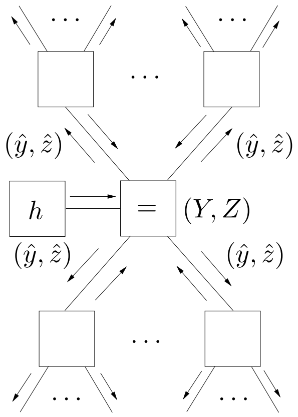
$$\mathrm{E}_f[g] \triangleq \sum_x f(x) g(x), \tag{4.54}$$

but the naive computation of (4.54) is not tractable. If it is "easy" to sample from f, one may generate a list samples $\{\hat{x}^{(i)}\}_{i=1}^N$ from f and evaluate the expectation (4.54) as

$$\mathrm{E}_f[g] \triangleq \frac{1}{N} \sum_{i=1}^N g\big(\hat{x}^{(i)}\big). \tag{4.55}$$



(a) Jointly sampling Y and Z



(b) Sampling the clustered variable (Y, Z)

Figure 4.17: Grouping in the context of Gibbs sampling; the variables X and Y are clustered.

Suppose now that sampling from f is "hard", and hence the approach (4.55) is not feasible. One may then draw samples $\big\{\hat{x}^{(i)}\big\}_{i=1}^{N}$ from a *different* function h with $\mathrm{supp}(f) \subseteq \mathrm{supp}(h)$, and compute (4.54) as

$$\mathrm{E}_f[g] \triangleq \frac{1}{N}\sum_{i=1}^{N} w^{(i)} g\big(\hat{x}^{(i)}\big), \tag{4.56}$$

where the weights w_i are given by

$$w^{(i)} \triangleq \frac{f\big(\hat{x}^{(i)}\big)}{h\big(\hat{x}^{(i)}\big)}. \tag{4.57}$$

The approach (4.56)–(4.57) is called importance sampling [171, p. 90–107]. We have applied this method in (4.35). Importance sampling is particularly natural when f factorizes, e.g.,

$$f(x) \triangleq f_1(x) f_2(x). \tag{4.58}$$

One may draw samples $\big\{\hat{x}^{(i)}\big\}_{i=1}^{N}$ ("importance samples") from f_1 and weight those samples by the function f_2 ("importance function"):

$$w^{(i)} \triangleq f_2\big(\hat{x}^{(i)}\big). \tag{4.59}$$

A message-passing view of this procedure is suggested in Fig. 4.18. The

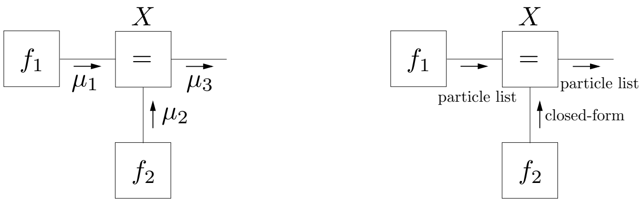


Figure 4.18: Importance sampling as message passing.

message μ_1 is represented by the list of samples $\big\{\hat{x}^{(i)}\big\}_{i=1}^{N}$ (with uniform weights). For the message μ_2, a closed-form expression is available, i.e., $\mu_2 \triangleq f_2$. The outgoing message μ_3 is the list of samples $\mu_3 \triangleq \big\{\hat{x}^{(i)}, w^{(i)}\big\}_{i=1}^{N}$, where $w^{(i)}$ is defined in (4.59). Importance sampling may be viewed as a particular instance of weighted sampling, where (1) the local node function g is an equality constraint node; (2) one of the messages is a list of samples; (3) the other message is available in closed-form.

Remark 4.4. (Importance sampling from f_1 or f_2)
If it is hard to *directly* sample from f_1 and f_2, one may use importance sampling (4.56) (4.57) to obtain (weighted) samples $\mu_1 \triangleq \big\{\hat{x}^{(i)}, \tilde{w}^{(i)}\big\}_{i=1}^{N}$ from f_1. The message μ_3 is represented as a list of samples $\big\{\hat{x}^{(i)}, w^{(i)}\big\}_{i=1}^{N}$, where (cf. (4.58))

$$w^{(i)} \triangleq \tilde{w}^{(i)} f_2\big(\hat{x}^{(i)}\big). \tag{4.60}$$

This method is the key to particle filtering, which is the subject of next section.

4.6.4 Particle Filtering

Particle filtering (or "sequential Monte-Carlo integration") stands for forward-only message passing in a state-space model of the form

$$f(s_0, s_2, \ldots, s_N, y_1, y_2, \ldots, y_N) \triangleq f_A(s_0) \prod_{k=1}^{N} f_A(s_{k-1}, s_k) f_B(s_k, y_k), \tag{4.61}$$

where (some of the) messages are represented by lists of samples [59] (see Fig. 4.19). More precisely, the messages μ_k and $\tilde{\mu}_k$ in Fig. 4.19 are

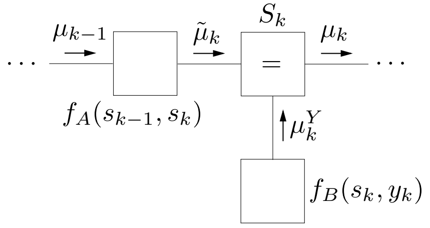


Figure 4.19: Particle filtering as message passing. One time slice of the state-space model is shown.

represented as lists of samples. In the basic particle filter, the list $\tilde{\mu}_k$ is obtained from μ_{k-1} by weighted sampling. The sampling-importance-resampling particle filter (SIR) uses unweighted sampling instead. In both particle filters, the list μ_k is generated from $\tilde{\mu}_k$ by importance sampling (cf. Fig. 4.18): the message $\tilde{\mu}_k$, μ_k^Y and μ_k in Fig. 4.19 correspond to the message μ_1, μ_2 and μ_3 respectively in Fig. 4.18.

Remark 4.5. (Smoothing by particle methods)
Along the same lines, particle methods can be used for smoothing in state-space models, as depicted in Fig. 4.20. The forward messages (μ_k^F and $\tilde{\mu}_k^F$), backward messages (μ_k^B and $\tilde{\mu}_k^B$), and upward messages μ_k^U are represented as lists of samples. The forward and backward messages are updated by applying the particle filter in a forward and backward sweep respectively. There are several ways to obtain the upward messages μ_k^U. One could generate them from the product of incoming particle lists $\tilde{\mu}_k^F$ and μ_k^B (by means of the methods of [88], see Section 4.6.1). We propose a simpler alternative based on importance sampling. One possible scheme is shown in Fig. 4.21(a). The importance samples are given by $\mu_k^B \triangleq \left\{\hat{s}_{B,k}^{(i)}, w_{B,k}^{(i)}\right\}_{i=1}^N$, and the importance function equals:

$$f(s_k) \quad \triangleq \quad \tilde{\mu}_k^F(s_k)\mu_k^Y(s_k), \tag{4.62}$$

where the message $\tilde{\mu}_k^F(s_k)$ is obtained by applying the sum-product rule to the node $f_A(s_{k-1}, s_k)$ with as incoming message the particle list $\mu_{k-1}^F \triangleq \left\{\hat{s}_{F,k-1}^{(i)}, w_{F,k-1}^{(i)}\right\}_{i=1}^N$ (cf. (4.35)), i.e.,

$$\tilde{\mu}_k^F(s_k) \quad \triangleq \quad \sum_{i=1}^N w_{F,k-1}^{(i)} f_A(\hat{s}_{F,k-1}^{(i)}, s_k). \tag{4.63}$$

The message μ_k^U is the list of samples $\mu_k^U \triangleq \left\{\hat{s}_{B,k}^{(i)}, w_{U,k}^{(i)}\right\}_{i=1}^N$, where

$$\begin{aligned} w_{k,U}^{(i)} \quad &\triangleq \quad w_{B,k}^{(i)} f(\hat{s}_{B,k}^{(i)}), & (4.64)\\ &\propto \quad w_{B,k}^{(i)} \mu_k^Y(\hat{s}_{B,k}^{(i)}) \sum_{j=1}^N w_{F,k-1}^{(j)} f_A(\hat{s}_{F,k-1}^{(j)}, \hat{s}_{B,k}^{(i)}). & (4.65) \end{aligned}$$

Obviously, one may exchange the roles of $\tilde{\mu}_k^F$ and μ_k^B, as depicted in Fig. 4.21(b): the importance samples are now given by $\tilde{\mu}_k^F$, and the importance function equals:

$$f(s_k) \quad \triangleq \quad \mu_k^B(s_k)\mu_k^Y(s_k), \tag{4.66}$$

where the message $\mu_k^B(s_k)$ is obtained by applying the sum-product rule to the node $f_A(s_k, s_{k+1})$ with as incoming message the particle list $\tilde{\mu}_{k+1}^B \triangleq \left\{\hat{s}_{B,k+1}^{(i)}, w_{B,k+1}^{(i)}\right\}_{i=1}^N$, i.e.,

$$\mu_k^B(s_k) \triangleq \sum_{i=1}^N w_{B,k+1}^{(i)} f_A(s_k, \hat{s}_{B,k+1}^{(i)}). \tag{4.67}$$

One may generate half of the samples by the method of Fig. 4.21(a), and the other half by the method of Fig. 4.21(b), which leads to a symmetrical situation.

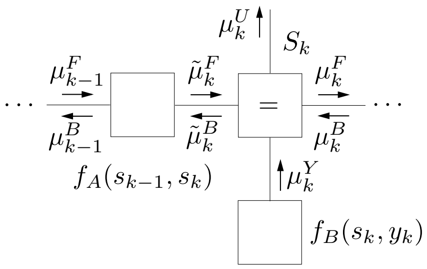


Figure 4.20: Smoothing by particle methods. One time slice of the state-space model is shown.

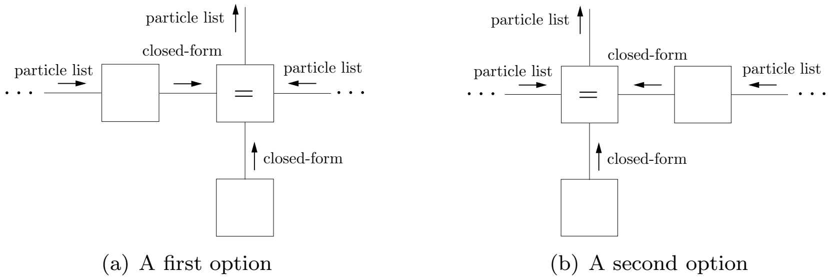


Figure 4.21: Computing μ_k^U by importance sampling.

Remark 4.6. (Particle filtering for constant parameters)
Particle filtering is not well-suited for inferring *constant* parameters. If s_k

is constant, i.e., $s_k \triangleq s$ and hence $f_A(s_{k-1}, s_k) \triangleq \delta(s_{k-1} - s_k)$ for all k, the (un)weighted sampling step is trivial: the messages $\tilde{\mu}_k$ are identical to the messages μ_{k-1}. Since importance sampling does not change the *position* of the particles, but only their *weights*, the position of the particles is in all lists μ_k identical. In other words, the particles do not "explore" the parameter space, and the resulting approximation of the posterior density of S is rather poor. Liu et al. [115] proposed a heuristic scheme to deal with this problem. The idea is to introduce "artificial evolution": the function $f_A(s_{k-1}, s_k) \triangleq \delta(s_{k-1} - s_k)$ is replaced by some conditional $q(s_k|s_{k-1})$, e.g., a Gaussian distribution with mean s_{k-1}. The (un)weighted sampling step is then no longer trivial: it modifies the position of the particles. However, artificial evolution introduces noise in the system: the lists of samples are "wider" than the true posterior density. Therefore, Liu et al. propose to move the particles towards the mean of the list after each (un)weighted sampling step ("shrinkage"). Shrinkage can exactly compensate for the increase in variance due to artificial evolution. However, it does not eliminate the distortion of the higher-order moments.

Remark 4.7. (Particle filters: pros and cons)
Both the basic and SIR particle filter suffer from certain problems. Generally speaking, due to the *repeated* sampling, small deviations from the true densities μ_k and $\tilde{\mu}_k$ will accumulate and eventually lead to drastically distorted sample lists. In particular, after several iterations of the basic particle filter, all but one particle have zero weight ("degeneracy"); in the SIR particle filter, all particles will eventually coincide ("sample impoverishment"). It is therefore recommendable to alternate weighted and unweighted sampling to generate the lists $\tilde{\mu}_k$ [59]. Despite sophisticated (heuristic) alternation schemes, the resulting sample lists may still be quite far from the true densities, especially when the latter are narrow (as in the high-SNR regime). In the literature, some other heuristic methods have been proposed to improve the basic and SIR particle filter, e.g., artificial evolution in conjunction with shrinkage [115] (cf. Remark 4.6). We have tried out most of the proposed schemes. The results of our experiments indicate that the standard particle methods are not recommendable if high-precision results are required, i.e., posterior densities or estimates. In fact, numerical integration often amounts to more accurate results; this method is however only applicable in low-dimensional systems.

4.6.5 Markov-Chain Monte-Carlo Methods (MCMC)

As in Section 4.6.3, we wish to draw samples from a probability function ("message") f from which it is hard to sample directly. As we explained in that section, samples from f may be obtained by importance sampling: one draws samples from a *different* function h and corrects for that by weighting the obtained samples. Markov-Chain Monte-Carlo methods (MCMC) are alternative sampling methods that are similar in spirit [171]. The idea is to sample repeatedly from an ergodic Markov chain with stationary distribution f. We now briefly present the most well-known MCMC method, i.e., the Metropolis-Hastings algorithm [171]. This algorithm is based on a conditional density $q(y|x)$ from which it is assumed to be easy to sample. In addition, q is supposed to be symmetric, i.e., $q(y|x) = q(x|y)$. Usually, the function q fully factorizes, i.e.,

$$q(y_1, \ldots, y_N | x_1, \ldots, x_N) \stackrel{\triangle}{=} \prod_{k=1}^{N} q(y_k|x_k). \tag{4.68}$$

An example of a function q is a Gaussian distribution with mean x and diagonal covariance matrix. The Metropolis-Hastings algorithm generates samples $\hat{x}$ from the "target" function f by the following iterative procedure:

a) Choose an initial value $\hat{x}$.

b) Sample $\hat{y}$ from $q(y|\hat{x})$.

c) Set
$$\hat{x} \stackrel{\triangle}{=} \hat{y} \quad \text{with probability } p \tag{4.69}$$
where
$$p \stackrel{\triangle}{=} \min\left\{\frac{f(\hat{y})}{f(\hat{x})}, 1\right\} \tag{4.70}$$

d) Iterate 2–3 a sufficient number of times.

Note that the function f must be available up to some constant.

Similarly as Gibbs sampling, the Metropolis-Hastings algorithm can be interpreted as a message-passing algorithm that operates on a factor graph of f. The following steps are iterated:

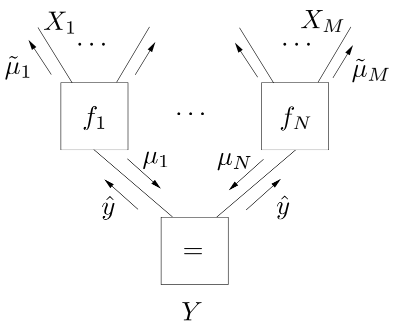


Figure 4.22: Application of the Metropolis-Hastings algorithm at a generic edge Y.

a) Select a variable (edge or equality constraint node) Y in the factor graph of f (see Fig. 4.22).

b) The edge Y generates the message $\hat{y}^{\text{new}}$ by sampling from $q(y|\hat{y})$.

c) Set $\hat{y} \triangleq \hat{y}^{\text{new}}$ with probability p where

$$p \triangleq \min\left\{\frac{f(\hat{y}^{\text{new}})}{f(\hat{y})}, 1\right\} \tag{4.71}$$

with

$$f(y) \triangleq \frac{\mu_1(y)\ldots\mu_N(y)}{\sum_y \mu_1(y)\ldots\mu_N(y)}. \tag{4.72}$$

The message $\hat{y}$ is broadcast to the neighboring nodes f_k ($k = 1, \ldots, M$).

d) The nodes f_k update their outgoing messages $\tilde{\mu}$ by applying the sum-product rule with as incoming messages the samples $\hat{y}$ and $\hat{x}_\ell$ ($\ell = 1, \ldots, M$) (cf. (4.84)).

One may group certain variables, as in Gibbs sampling (cf. Remark 4.3).

In the previous sections, we encountered several situations where we needed to draw samples from a probability function ("message"). As we explained in Section 4.6.1, sampling from a generic sum-product message $\mu_{g\to X}$ (4.36) requires samples from $g(x, \hat{z}_1, \ldots, \hat{z}_N)$; if sampling from g is hard, and importance sampling is for some reason inconvenient, one may resort to the Metropolis-Hastings algorithm.

MCMC can also be applied in the context of Gibbs sampling, more specifically, to generate samples from $f(y)$ (cf. (4.43)). The resulting algorithms are in the literature referred to as hybrid MCMC [171, pp. 392–396].

MCMC can be used in the context of particle filters too [20] [97]. In the case of filtering, the message μ_k is then not generated from $\tilde{\mu}_k$ by importance sampling (cf. Fig. 4.19), instead it is obtained by means of the Metropolis-Hastings algorithm with target function f:

$$f(s_k) \quad \stackrel{\triangle}{=} \quad \mu_k^Y(s_k)\tilde{\mu}_k^F(s_k), \tag{4.73}$$

where $\tilde{\mu}_k^F(s_k)$ is given by (4.63). As a consequence

$$f(s_k) \quad \stackrel{\triangle}{=} \quad \mu_k^Y(s_k)\sum_i w_{k-1}^{(i)} f_A(\hat{s}_{k-1}^{(i)}, s_k). \tag{4.74}$$

One may proceed similarly in the case of smoothing. This combined MCMC-particle filtering approach is less prone to problems as sample impoverishment and degeneracy, but it is computationally complex.

4.6.6 Simulated Annealing

The original simulated annealing algorithm is an extension of the Metropolis-Hastings algorithm [171, pp. 163–169]. It can be used:

- to sample from a multivariate function $f(x_1, \ldots, x_N)$,
- to find the mode of the function f.

The key idea is to draw samples from functions f^α, where the (positive) exponent α *increases* in the course of the algorithm. The initial value of α is close to zero (e.g., $\alpha = 0.1$). If one wishes to obtain samples from f, one halts the algorithm as soon as $\alpha = 1$. If one tries to find the mode of α, the end value of α is much larger (e.g., $\alpha = 10$ or 100). Note that for small values of α (i.e., $0 \leq \alpha < 1$), the function f^α is flatter than the target function f, whereas for large values of α (i.e., $\alpha \gg 1$), the function f^α mainly consists a narrow peak centered at the global maximum of f.

Simulated annealing works as follows:

a) Choose an initial value $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$.

b) Choose an initial value α (e.g., $\alpha = 0.1$).

c) Sample a new value $\hat{y}$ from $q(y|\hat{x})$.

d) Set $\hat{x} \stackrel{\triangle}{=} \hat{y}$ with probability p, where

$$p \stackrel{\triangle}{=} \min\left\{ \left(\frac{f(\hat{y})}{f(\hat{x})} \right)^{\alpha}, 1 \right\} \tag{4.75}$$

e) Iterate 3–4 a "large" number of times.

f) Increase α according to some schedule.

g) Iterate 5–6 until convergence or until the available time is over.

Various stopping criteria and several schemes to increase α are available in the literature.[4]

The principle of simulating annealing is generic. It can be applied to *any* message-passing algorithm, not only to the Metropolis-Hastings algorithm.[5] The idea is to replace a local function f by a power f^α.[6] In the course of the message-passing algorithm, α increases. If one is interested in posterior probabilities, the algorithm is halted as soon as $\alpha = 1$. If one tries to find the mode of a function, one stops the algorithm at a larger value of α.

[4]We refer to [171, pp. 163–169] for detailed information and numerous references.

[5]If it is interleaved with a particle method (such as Metropolis-Hastings), it is called "stochastic annealing"; otherwise, it is called "deterministic annealing".

[6]This only makes sense if the function f is "soft", i.e., if it is not a Dirac delta.

4.6.7 Application

Constant-Phase Model

The forward messages μ_k^F in Fig. 4.23 are represented as lists of samples $\mu_k^F \triangleq \{\hat{\theta}_k^{(i)}\}_{i=1}^N$, whereas the downward messages $\mu_{g\to\Theta_k}$ are available in closed-form (cf. (4.23)). The messages in Fig. 4.23 are computed as follows:

① Initialize the list μ_1^F with samples drawn from $\mu_{g\to\Theta_1}$.

② The particle lists μ_k^F are updated in a forward sweep by means of importance sampling combined with the evolution/shrinkage method of Liu et al. [115] (cf. Remark 4.6).

③ The upward messages $\mu_{\boxtimes\to X_k}(x_k)$ are computed as weighted averages over the list μ_L^F:

$$\mu_{\boxtimes\to X_k}(x_k) \overset{\triangle}{\propto} \sum_{i=0}^{N-1} w_L^{(i)} \mu_{Z_k\to\boxtimes}\big(x_k \exp\big(j\hat{\theta}_L^{(i)}\big)\big), \tag{4.76}$$

where $w_L^{(i)}$ is the weight of particle $\hat{\theta}_L^{(i)}$.

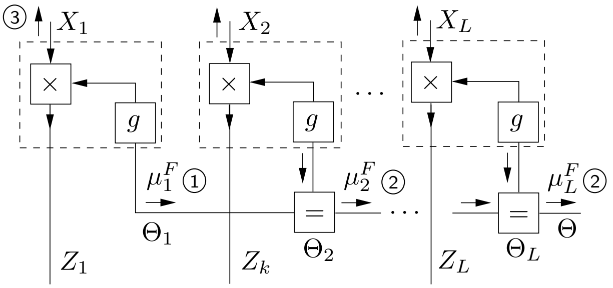


Figure 4.23: Particle filtering in the factor graph of the constant-phase model.

Random-Walk Phase Model

The forward, backward, and upward messages $\mu_{p_k \to \Theta_{k-1}}$, $\mu_{p_k \to \Theta_k}$ and $\mu_{\Theta_k \to g}$ respectively are lists of samples, the downward messages $\mu_{g \to \Theta_k}$ are available in closed-form (see Fig. 4.24). The messages in Fig. 4.24 are updated as follows:

① Initialize the lists $\mu_{\Theta_1 \to p_2}$ and $\mu_{\Theta_L \to p_L}$ with samples drawn from $\mu_{g \to \Theta_1}$ and $\mu_{g \to \Theta_L}$ (cf. (4.23)).

② The messages $\mu_{p_k \to \Theta_k}$ and $\mu_{p_k \to \Theta_{k-1}}$ are updated in a forward and backward sweep respectively. At the phase noise nodes, sample lists are generated by (un)weighted sampling; this is implemented by simply adding "phase noise" to the incoming particles (cf. (4.2)). At the equality constraint nodes, importance sampling is applied (cf. Section 4.6.4).

③ The messages $\mu_{\Theta_k \to g}$ $(1 < k < L)$ are obtained by importance sampling (cf. Remark 4.5).

④ The messages $\mu_{\boxtimes \to X_k}(x_k)$ are evaluated as weighted averages over the lists $\mu_{\Theta_k \to g} \overset{\triangle}{=} \{\hat{\theta}_k^{(i)}, w_k^{(i)}\}_{i=1}^N$:

$$\mu_{\boxtimes \to X_k}(x_k) \overset{\triangle}{\propto} \sum_{i=0}^{N-1} w_k^{(i)} \mu_{Z_k \to \boxtimes}\big(x_k \exp(j\hat{\theta}_k^{(i)})\big). \qquad (4.77)$$

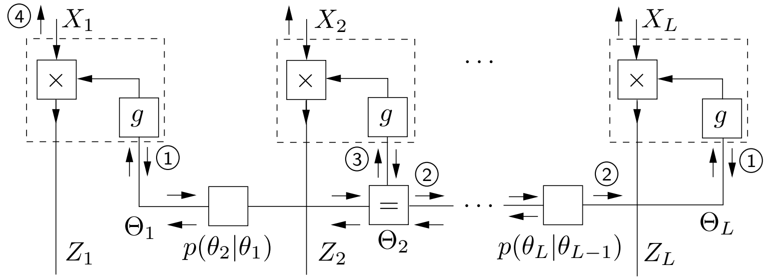


Figure 4.24: Particle filtering in the factor graph of the random-walk phase model.

4.6.8 Summary

We summarize the main ideas of this section.

- In particle methods, probability distributions ("messages") are represented as **lists of samples**. Integrals are evaluated as **averages** over lists of samples.

- We have given a **local view** of particle methods, i.e., we have shown how particle methods can be applied at particular **nodes** in the factor graph of the system at hand. In particular, we have explained how samples can be drawn from a **generic sum-product message**. We treated **discrete** and **continuous** variables. We considered various representations for the **incoming messages**, i.e., lists of samples, quantized messages, hard decisions, Gaussian distributions etc. Several combinations of incoming messages amount to **existing algorithms**, others to **novel algorithms**.

- In this setting, particle methods can straightforwardly be **combined** with other methods, such as gradient methods, decision-based methods (as ICM), Kalman-style algorithms (based on Gaussian distributions; see Appendix H) etc. In addition, such combinations can **systematically** be explored.

- We have shown how standard Monte-Carlo methods can be understood from this local viewpoint:

  - In **Gibbs sampling**, messages are represented by a **single sample**.
  - **Importance sampling** can be viewed as message passing at an **equality constraint node**, where one incoming message is available in closed form, the other is a particle list. Also the outgoing message is represented by a particle list.
  - **Particle filtering** stands for forward/backward message passing on the factor graph of a **state-space model**, where the forward/backward messages are lists of samples.
  - **MCMC** is a method to draw samples from a "complicated" probability function ("message"). It may be used in combination with Gibbs sampling and particle filtering.

– **Simulated Annealing** was originally formulated as a particle method. However, simulated annealing is a generic idea, and it can be applied to any message-passing algorithm: certain factors (in the factor graph of the system at hand) are raised to a positive **power** α; in the course of the message-passing algorithm, the power α increases.

4.7 Adaptive Quantization

4.7.1 General Idea

In numerical integration and particle methods, integrals are replaced by finite sums (cf. (4.27) and (4.35)). In numerical integration, the quantization levels are uniform, whereas in particle filtering, the quantization levels are sampled from a distribution, hence, they are non-uniform. Obviously, non-uniform quantization (and in particular particle methods) could in principle lead to better results than uniform quantization. We pointed out before, however, that for the particular problem of estimation in state-space models, particle methods (i.e., particle filtering), seems to perform *worse* than numerical integration, mainly due to the recursive sampling (cf. Remark 4.7). We propose an alternative (heuristic) method to obtain non-uniform quantization levels, which in particular situations leads to better results than uniform quantization (numerical integration). The idea is simple: where the function $f(x)$ attains large values, the density of quantization levels should be large (cf. Fig. 4.13 (left)). In contrast to particle filtering, we do not try to achieve this goal by sampling from $f(x)$, but by *shrinkage* (see Fig. 4.25): the quantization levels are recursively moved towards the mean (or maximum or median). Suppose an initial quantization $\{x^{(i)}\}_{i=1}^{N}$ of x is given. Shrinkage stands for the iterative application of the following two steps:

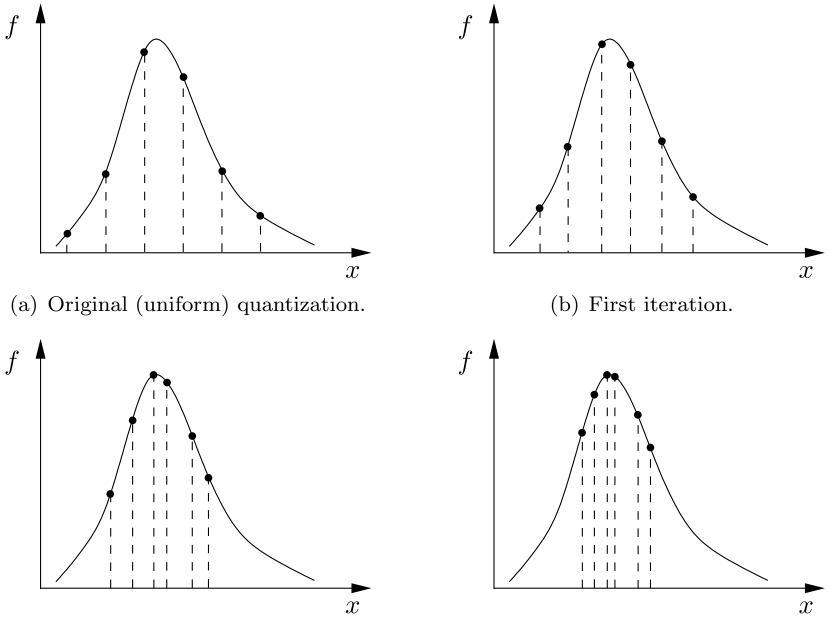
a) Compute the mean $\bar{x}$:

$$\bar{x} \triangleq \frac{1}{N}\sum_{i=1}^{N} f\big(x_{\text{old}}^{(i)}\big)\, x_{\text{old}}^{(i)}. \tag{4.78}$$

b) Move the quantization levels towards the mean:

$$x_{\text{new}}^{(i)} \triangleq (1-\varepsilon)x_{\text{old}}^{(i)} + \varepsilon\bar{x}, \tag{4.79}$$

where ε is a small positive number that may depend on the iteration number. In Fig. 4.25, three iterations of this algorithm are depicted. The resulting quantization levels are obviously not *samples* from f. The



(a) Original (uniform) quantization. (b) First iteration.

(c) Second iteration. (d) Third iteration.

Figure 4.25: Adaptive quantization by shrinkage.

method works only well for unimodal distributions. If f is multimodal, one may apply a clustering algorithm first, and apply shrinkage in each cluster. Note that one may use the maximum or median rather than the mean (4.78).

Shrinkage can be integrated in a generic message-passing algorithm that operates on a factor graph as follows:

a) Initialize the quantization levels of all quantized variables x in the factor graph. The quantization levels may be uniform; alternatively, they may be generated by sampling (particle list).

b) Initialize all messages.

c) Update all messages in the graph according to the sum-product rule or a suitable approximation. In the case of quantized variables, the sum-product rule is evaluated as in (4.27).

d) Compute the marginals $f(x)$ of the quantized variables x.

e) Shrink the quantization levels of x based on the marginals $f(x)$ (cf. (4.78) and (4.79)).

f) Iterate 3–5.

4.7.2 Application

We extend the phase estimators based on numerical integration (cf. Section 4.5.2) with shrinkage.

Constant-Phase Model

Shrinkage is applied between step ② (computation of μ_k^F) and ③ (computation of $\mu_{\boxtimes \to X_k}$):

a)

$$\bar{\theta} = \arg \sum_{i=0}^{N-1} \mu_L^F(\hat{\theta}_{\text{old}}^{(i)}) \exp\left(j\hat{\theta}_{\text{old}}^{(i)}\right), \tag{4.80}$$

b)

$$\hat{\theta}_{\text{new}}^{(i)} = \arg \left[(1-\varepsilon)\exp\left(j\hat{\theta}_{\text{old}}^{(i)}\right) + \varepsilon \exp(j\bar{\theta})\right], \tag{4.81}$$

where $i = 0, 1, \ldots, N-1$.

The update ② and the shrinkage step (4.80)–(4.81) are iterated a number of times before the update ③ is carried out.

Random-Walk Phase Model

Shrinkage is applied between step ① (computation of $\mu_{\Theta_k \to p_k}$ and $\mu_{\Theta_k \to p_{k-1}}$) and ② (computation of $\mu_{\Theta_k \to g}$):

a)

$$\bar{\theta} = \arg \sum_{i=0}^{N-1} \mu_{p_{k+1} \to \Theta_k}(\hat{\theta}_{\text{old}}^{(i)}) \mu_{p_k \to \Theta_k}(\hat{\theta}_{\text{old}}^{(i)}) \exp\big(j\hat{\theta}_{\text{old}}^{(i)}\big), \tag{4.82}$$

b)

$$\hat{\theta}_{\text{new}}^{(i)} = \arg \Big[(1-\varepsilon)\exp\big(j\hat{\theta}_{\text{old}}^{(i)}\big) + \varepsilon \exp(j\bar{\theta})\Big], \tag{4.83}$$

where $i = 0, 1, \ldots, N-1$.

The update ①, the shrinkage step (4.82)–(4.83), and the update ② are iterated a number of times before the update ③ is carried out.

4.7.3 Summary

- Adaptive quantization is a simple heuristic method to generate **non-uniform** quantization levels.
- It can easily be **integrated** in a message-passing algorithm with quantized messages.

4.8 Gradient Methods

In this section, we describe steepest descent (or "steepest ascent" or "gradient descent") as a message-passing algorithm. The results in this section are based on joint work with Sascha Korl [46].

4.8.1 General Idea

It is common to represent a probability function $f(x)$ by a single value such as:

- its mode $\hat{x}^{\max} \triangleq \operatorname{argmax}_x [f(x)]$
- its mean
- its median
- a sample from f (as in Gibbs sampling).

The probability function f is then approximated by the Dirac delta $\delta(x - \hat{x})$, as illustrated in Fig. 4.26 for $\hat{x} = \hat{x}^{\max}$.

In particular, if sum-product messages are represented by single values, the integral-product rule is approximated as follows.

Integral-product rule evaluated by means of hard decisions:

$$\mu_{f \to Y}(y) \quad \overset{\triangle}{\propto} \quad f(y, \hat{x}_1, \ldots, \hat{x}_N), \tag{4.84}$$

where $\hat{x}_k$ is a hard estimate of X_k, representing the message $\mu_{X_k \to f}$.

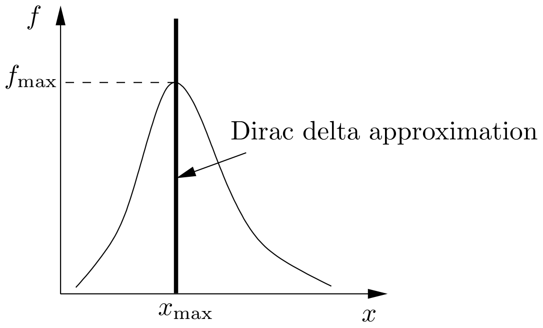


Figure 4.26: A probability density function and its approximation by a Dirac delta located at its mode $x^{\max}$.

In the rest of this section, we focus on the mode $\hat{x}^{\max} \triangleq \operatorname{argmax}_x [f(x)]$. Let us have a look at a simple example.

Example 4.2. (Mode of a message)
We consider the following model:

$$Y_k \triangleq e^{j\Theta} + N_k, \tag{4.85}$$

where N_k is complex white Gaussian noise with (known) variance $2\sigma_N^2$, i.e., σ_N^2 per dimension. We wish to estimate the unknown phase $\Theta \in [0, 2\pi)$ from N observations $y \triangleq (y_1, \ldots, y_N)$. The factor graph of Fig. 4.27 depicts the conditional probability density function $p(y|\theta)$. The factors $p(y_k|\theta_k)$ are defined as

$$p(y_k|\theta_k) \triangleq \frac{1}{2\pi\sigma_N^2} e^{-|y_k - e^{j\theta_k}|^2/2\sigma_N^2}. \tag{4.86}$$

Since the factor graph is cycle-free, sum-product message passing leads to the exact marginals. The messages $\mu\!\uparrow\!(\theta_k)$ are given as

$$\begin{aligned} \mu\!\uparrow\!(\theta_k) &= p(y_k|\theta_k) & (4.87)\\ &= \frac{1}{2\pi\sigma_N^2} e^{-|y_k - e^{j\theta_k}|^2/2\sigma_N^2}. & (4.88)\end{aligned}$$

The message $\mu\!\uparrow\!(\theta)$ equals

$$\begin{aligned} \mu\!\uparrow\!(\theta) &= \prod_{k=1}^{N} \mu_k\!\uparrow\!(\theta) & (4.89)\\ &= \prod_{k=1}^{N} \frac{1}{2\pi\sigma_N^2} e^{-|y_k - e^{j\theta}|^2/2\sigma_N^2}. & (4.90)\end{aligned}$$

The ML-estimate of Θ is the mode of $\mu\!\uparrow\!(\theta) = p(y|\theta)$. Since the function $p(y_k|\theta)$ is concave, the ML-estimate $\hat{\theta}^{\mathrm{ML}}$ is obtained by solving the equation:

$$\left.\frac{d\mu\!\uparrow\!(\theta)}{d\theta}\right|_{\hat{\theta}^{\mathrm{ML}}} \overset{!}{=} 0. \tag{4.91}$$

The (unique) solution of (4.91) is

$$\hat{\theta}^{\mathrm{ML}} = \arg\left(\frac{\sum_{k=1}^{N} \mathrm{Im}[y_k]}{\sum_{k=1}^{N} \mathrm{Re}[y_k]}\right). \tag{4.92}$$

□

In the previous example, we obtained a closed-form expression for the mode. In many practical situations, there is no closed-form expression for the mode available. One may then apply numerical methods such as gradient methods. The familiar steepest descent (or “steepest ascent”

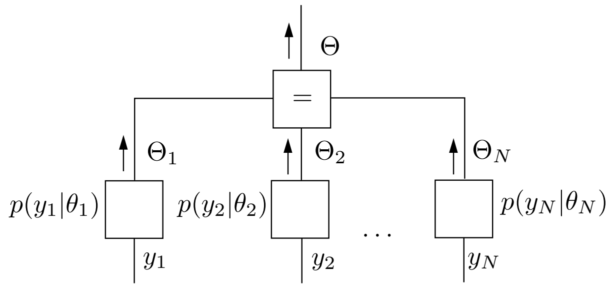


Figure 4.27: ML estimation of the phase Θ from non-modulated observations.

or "gradient descent/ascent") method tries to find $\hat{\theta}_{\max}$ as follows [19]: starting from some initial guess $\hat{\theta}^{(0)}$, iterate the rule:

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \nabla_\theta f(\theta)|_{\hat{\theta}^{(k)}}, \tag{4.93}$$

for $k = 1, 2, 3, \ldots$, where the parameter λ_k (the "step size") is a positive real number that may depend on k. An alternative update rule is

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \nabla_\theta \log f(\theta)|_{\hat{\theta}^{(k)}}. \tag{4.94}$$

The update rule (4.93) or (4.94) is iterated until a fixed point is reached or until the available time is over. Note that rule (4.94) is often preferable to (4.93) if $g(\theta)$ strongly varies, since the logarithm in (4.94) compresses the range of $g(\theta)$.

In this section, we investigate how steepest descent can be applied in the context of the sum-product algorithm. We start by considering the factor graph depicted in Fig. 4.28(a), which represents the global function $f(\theta) \stackrel{\triangle}{=} f_A(\theta)f_B(\theta)$. The gradient $\nabla_\theta f(\theta)$ in update rule (4.93) is given by

$$\nabla_\theta f(\theta) \quad = \quad f_B(\theta)\nabla_\theta f_A(\theta) + f_A(\theta)\nabla_\theta f_B(\theta), \tag{4.95}$$

and similarly, the gradient $\nabla_\theta \log f(\theta)$ in update rule (4.94) equals

$$\nabla_\theta \log f(\theta) \quad = \quad \nabla_\theta \log f_A(\theta) + \nabla_\theta \log f_B(\theta). \tag{4.96}$$

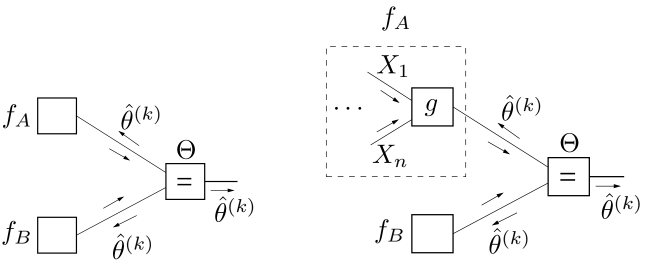
Steepest descent according to rule (4.94) (and similarly (4.93)) may be viewed as follows:

a) The equality constraint node in Fig. 4.28(a) broadcasts the estimate $\hat{\theta}^{(k)}$. Node f_A replies with the message $\nabla_\theta \log f_A(\theta)|_{\theta^{(k)}}$ and likewise node f_B.

b) A new estimate $\hat{\theta}^{(k+1)}$ is computed as

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \Big(\nabla_\theta \log f_A(\theta)\Big|_{\hat{\theta}^{(k)}} + \nabla_\theta \log f_B(\theta)\Big|_{\hat{\theta}^{(k)}} \Big). \tag{4.97}$$

c) Iterate 1–2.



(a) f_A and f_B as closed boxes.

(b) Opening the box f_A.

Figure 4.28: Factor graph of $f(\theta) = f_A(\theta) f_B(\theta)$.

In Fig. 4.28(a), the nodes f_A and f_B may be summaries of the subgraph "behind" them, as illustrated in Fig. 4.28(b): the function f_A is a summary of the dashed box. This box contains a.o. the local node g, which is connected to the equality constraint node Θ. The summary $f_A(\theta)$ is computed from the messages $\mu_{X_k \to g}$, arriving at the node g from the left, according to the sum-product rule:

$$f_A(\theta) \propto \sum_{x_1,\ldots,x_n} g(x_1,\ldots,x_n,\theta) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell). \tag{4.98}$$

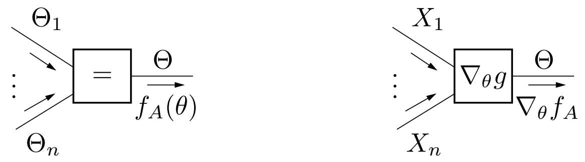
The above gradient method requires $\nabla_\theta f_A(\theta)$ and $\nabla_\theta \log f_A(\theta)$ (cf. (4.97)). In the following, we show how these expressions can be computed. We distinguish three cases:

a) g is an equality constraint node

b) g is differentiable

c) g corresponds to a deterministic mapping.

Equality constraint node


Θ_1
$=$
Θ
$\overrightarrow{f_A(\theta)}$
Θ_n
X_1
$\nabla_\theta g$
Θ
$\overrightarrow{\nabla_\theta f_A}$
X_n

(a) Equality constraint node. (b) Differentiable node function.

Figure 4.29: Generic nodes.

If g is an equality constraint node (see Fig. 4.29(a)), the required gradients are computed similarly as in (4.95) and (4.96):

$$\nabla_\theta f_A(\theta) = \sum_{\ell=1}^{n} \nabla_\theta \mu_{\theta_\ell \to \boxed{=}}(\theta) \prod_{m=1; m \neq \ell}^{n} \mu_{\theta_m \to \boxed{=}}(\theta), \tag{4.99}$$

and

$$\nabla_\theta \log f_A(\theta) = \sum_{\ell=1}^{n} \nabla_\theta \log \mu_{\Theta_\ell \to \boxed{=}}(\theta). \tag{4.100}$$

Differentiable node function

Let $g(x_1, \ldots, x_n, \theta)$ be differentiable w.r.t. θ. The gradient $\nabla_\theta f_A(\theta)$ can then be computed as follows:

$$\nabla_\theta f_A(\theta) \propto \sum_{x_1,\ldots,x_n} \nabla_\theta g(x_1, \ldots, x_n, \theta) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell). \tag{4.101}$$

Note that in (4.101), we differentiated under the integral sign; we will always assume in this thesis that this operation is allowed.[7] The update rule (4.101) can be viewed as applying the sum-product rule to the node $\nabla_\theta g$, as illustrated in Fig. 4.29(b). The incoming messages are the standard sum-product summaries $\mu_{X_\ell \to g}$. In other words, if g is differentiable, the differentiation operator does not propagate to the subgraph

[7]Appendix I lists necessary conditions for differentiation under the integral sign.

on the left of g; it is (only) applied to the local node function g. This is not the case if g corresponds to a deterministic mapping, which is the subject of next subsection.

The gradient $\nabla_\theta \log f_A(\theta)$ equals

$$\nabla_\theta \log f_A(\theta) \quad = \quad \frac{\nabla_\theta f_A(\theta)}{f_A(\theta)}, \tag{4.102}$$

and is computed from (4.98) and (4.101). In order to evaluate $\nabla_\theta \log f_A(\theta)$, the sum-product rule is applied both to g and $\nabla_\theta g$.

If the variables X_ℓ are discrete (and the alphabet is not "too large"), the expressions (4.101) and (4.102) can be evaluated in a straightforward manner. If on the other hand those variables (or a subset of them) are continuous, the integrals in (4.98) and (4.101) may be evaluated in several ways.

- In some cases, a closed-form expression of (4.98) or (4.101) exists.
- The integrals in (4.98) and (4.101) can be approximated based on canonical distributions as for example Gaussian distributions.
- The messages $\mu_{X_\ell \to g}(x_\ell)$ may be quantized messages with quantization levels $x_\ell^{(i_\ell)}$, then (cf. (4.27))

$$\nabla_\theta f_A(\theta) \tag{4.103}$$

$$\propto \sum_{i_1,\ldots,i_n} \nabla_\theta g\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big) \prod_{k=1}^{n} \mu_{x_k \to g}\big(x_k^{(i_k)}\big), \tag{4.104}$$

and

$$\nabla_\theta \log f_A(\theta) \tag{4.105}$$

$$= \frac{\sum\limits_{i_1,\ldots,i_n} \nabla_\theta g\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big) \prod_{k=1}^{n} \mu_{x_k \to g}\big(x_k^{(i_k)}\big)}{\sum\limits_{i_1,\ldots,i_n} g\big(x_1^{(i_1)},\ldots,x_n^{(i_n)},\theta\big) \prod_{k=1}^{n} \mu_{x_k \to g}\big(x_k^{(i_k)}\big)}, \tag{4.106}$$

where the sums are taken over the quantization levels (which may be different for each variable).

- The messages $\mu_{X_\ell \to g}(x_\ell)$ may be lists of samples $\{x_\ell^{(i_\ell)}\}$ (cf. (4.35)). Consequently

$$\nabla_\theta f_A(\theta) \propto \sum_{i_1,\ldots,i_n} \nabla_\theta g\big(x_1^{(i_1)}, \ldots, x_n^{(i_n)}, \theta\big), \tag{4.107}$$

  and

$$\nabla_\theta \log f_A(\theta) = \frac{\sum_{i_1,\ldots,i_n} \nabla_\theta g\big(x_1^{(i_1)}, \ldots, x_n^{(i_n)}, \theta\big)}{\sum_{i_1,\ldots,i_n} g\big(x_1^{(i_1)}, \ldots, x_n^{(i_n)}, \theta\big)}, \tag{4.108}$$

  where the sums are taken over the lists of samples.

- The incoming messages $\mu_{X_\ell \to g}(x_\ell)$ may be a hard decision $\hat{x}_\ell$. The expressions (4.101) and (4.102) reduce to (cf. (4.84))

$$\nabla_\theta f_A(\theta) \propto \nabla_\theta g(\hat{x}_1, \ldots, \hat{x}_n, \theta) \tag{4.109}$$

  and

$$\nabla_\theta \log f_A(\theta) = \frac{\nabla_\theta g(\hat{x}_1, \ldots, \hat{x}_n, \theta)}{g(\hat{x}_1, \ldots, \hat{x}_n, \theta)}. \tag{4.110}$$

- Combinations of the previous options are possible.

Deterministic mapping

We consider the case where the local function g corresponds to the deterministic mapping $y \triangleq h(x_1, \ldots, x_n, \theta)$, i.e.,

$$g(x_1, \ldots, x_n, y, \theta) \triangleq \delta\big(y - h(x_1, \ldots, x_n, \theta)\big). \tag{4.111}$$

We assume that h is differentiable. Let $v \triangleq (x_1, \ldots, x_n)$. The message $f_A(\theta)$ is computed as follows

$$f_A(\theta) \propto \sum_{x_1,\ldots,x_n} \delta\big(y - h(v, \theta)\big) \mu_{Y \to g}(y) \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell) \tag{4.112}$$

$$= \sum_{v} \mu_{Y \to g}\big(h(v, \theta)\big) \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell). \tag{4.113}$$

As a consequence

$$\nabla_\theta f_A(\theta) \propto \sum_v \nabla_\theta \big[\mu_{Y\to g}\big(h(v,\theta)\big)\big] \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell) \tag{4.114}$$

$$= \sum_v \nabla_\theta h(v,\theta)\ \nabla_y\, \mu_{Y\to g}(y)\big|_{y=h(v,\theta)} \cdot \prod_{\ell=1}^{n} \mu_{X_\ell \to g}(x_\ell). \tag{4.115}$$

Eq. (4.115) may be viewed as applying the sum-product rule to the node function $\nabla_\theta h$, as illustrated in Fig. 4.30. Besides the standard sum-product messages $\mu_{X_\ell\to g}(x_\ell)$, also the message $\nabla_y\, \mu_{Y\to h}(y)|_{h(v,\theta)}$ is required, which is the *gradient* of a sum-product message. The message $\nabla_y\, \mu_{Y\to g}(y)|_{h(v,\theta)}$ is computed by the same rules as $\nabla_\theta f_A(\theta)$ (cf. (4.99) (4.101) (4.115)). Similarly as in (4.101) and (4.102), the update rule (4.115) can be evaluated in several ways, depending on the data-type of the incoming messages. For example, if the incoming messages $\hat{x}_\ell$ and $\nabla_y\, \mu_{Y\to g}(y)|_{h(\hat{x},\theta)}$ are hard decisions, where $\hat{x}$ stands for $(\hat{x}_1, \ldots, \hat{x}_n)$, then

$$\nabla_\theta f_A(\theta) \quad \propto \quad \nabla_\theta g(\hat{v},\theta)\ \nabla_y\, \mu_{Y\to g}(y)\big|_{h(\hat{v},\theta)}. \tag{4.116}$$

This rule may be familiar to the reader who has some background in neural network theory: indeed, if one applies the rule (4.116) together with (4.97) (4.109) and (4.110) to a factor graph that represents a feed-forward neural network, one obtains the well-known backpropagation algorithm [22]. We refer to Appendix E.3 for more details.

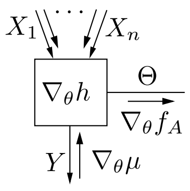


Figure 4.30: Deterministic mapping h.

We explain now how two well-known algorithms can be viewed as instances of the above generic rules: iterative conditional modes and stochastic approximation.

4.8.2 Iterative Conditional Modes

Suppose we wish to maximize a multivariate function $f(x_1, \ldots, x_N)$. This can be done by the following iterative algorithm called *Iterative Conditional Modes* (ICM) (or "cyclic maximization" or "alternating maximization" or "coordinate descent"):

a) Choose an initial value $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$.

b) Choose an index k.

c) Compute $\hat{x}_k \stackrel{\triangle}{=} \operatorname{argmax}_{x_k} f(x_k)$, where

$$f(x_k) \stackrel{\triangle}{=} \frac{f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)}{\sum_{x_k} f(\hat{x}_1, \ldots, \hat{x}_{k-1}, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)}. \tag{4.117}$$

d) Iterate 2–3 a "large" number of times.

Under the assumption that f is bounded above, ICM converges to a local maximum of f. Note that ICM is strongly related to Gibbs sampling (see Section 4.6.2). In Gibbs sampling, one draws a sample from $f(x_k)$, whereas in ICM one computes the mode of $f(x_k)$.

If it is hard to compute the mode $\hat{x}_k \stackrel{\triangle}{=} \operatorname{argmax}_{x_k} f(x_k)$ (Step c), one may select a value $\hat{x}_k^{\text{new}}$ which increases $f(x_k)$, i.e., $f(\hat{x}_k^{\text{new}}) \geq f(\hat{x}_k^{\text{old}})$ ("generalized ICM"). The value $\hat{x}_k^{\text{new}}$ may be obtained by gradient methods. Generalized ICM usually leads to a stationary point of f, but not necessarily to a local maximum.

ICM can readily be formulated as a message-passing algorithm:

a) Select a variable (edge or equality constraint node) Y in the factor graph of f (see Fig. 4.16).

b) The edge Y generates the message $\hat{y} \stackrel{\triangle}{=} \operatorname{argmax}_y f(y)$, where

$$f(y) \stackrel{\triangle}{=} \frac{\mu_1(x) \ldots \cdot \mu_N(y)}{\sum_y \mu_1(y) \ldots \mu_N(y)}, \tag{4.118}$$

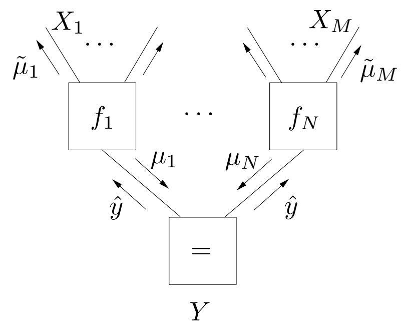


Figure 4.31: ICM at a generic edge Y.

and broadcasts $\hat{y}$ to its neighboring nodes f_k ($k = 1, \ldots, M$).

c) The nodes f_k update their outgoing messages $\tilde{\mu}$ by applying the sum-product rule with as incoming messages the samples $\hat{y}$ and $\hat{x}_\ell$ ($\ell = 1, \ldots, M$).

In the case of modified ICM, one replaces the message $\hat{y} \triangleq \operatorname{argmax}_y f(y)$ (Step c) by a value $\hat{y}$ which increases $f(y)$. If this value is determined by steepest descent, one obtains a similar situation as in Fig. 4.32. As in Gibbs sampling, one may group several variables (cf. Remark 4.3).

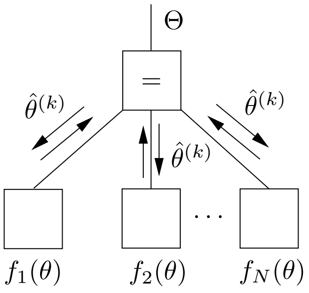
4.8.3 Stochastic Approximation

Suppose we wish to find the mode $\theta^{\max}$ of the global function $f(\theta) \triangleq \prod_{\ell=1}^{N} f_\ell(\theta)$ by means of steepest descent (see Fig. 4.32). Straightforward application of the generic rules of Section 4.8.1 (cf. Fig. 5.1) leads to the following message-passing algorithm:

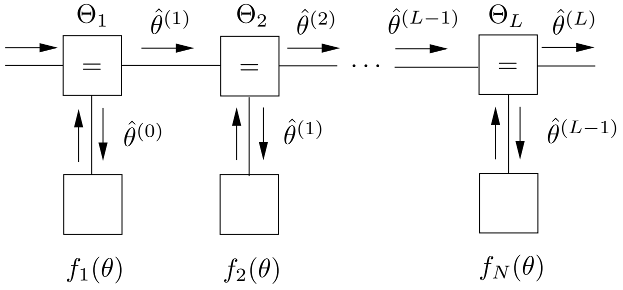
a) The equality constraint node in Fig. 4.32(a) broadcasts the estimate $\hat{\theta}^{(k)}$. The nodes f_ℓ reply with the message $\nabla_\theta \log f_\ell(\theta)|_{\theta^{(k)}}$.

b) A new estimate $\hat{\theta}^{(k+1)}$ is computed as

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} + \lambda_k \Big[\sum_{\ell=1}^{N} \nabla_\theta \log f_\ell(\theta) \Big|_{\hat{\theta}^{(k)}} \Big]. \tag{4.119}$$



(a) Standard scheduling.



(b) Stochastic approximation.

Figure 4.32: Gradient methods for estimating Θ.

c) Iterate 1–2.

An alternative scheme is depicted in Fig. 4.32(b). The following steps are performed in a forward sweep ($k = 1, \dots, N$):

a) The equality constraint node Θ_k sends the estimate $\hat{\theta}^{(k-1)}$ to the node f_k. The latter replies with the message $\nabla_\theta \log f_k(\theta)|_{\hat{\theta}^{(k-1)}}$.

b) A new estimate $\hat{\theta}^{(k)}$ is computed as

$$\hat{\theta}^{(k)} = \hat{\theta}^{(k-1)} + \lambda_k \nabla_\theta \log f_k(\theta)\Big|_{\hat{\theta}^{(k-1)}}, \tag{4.120}$$

which is then sent to the equality constraint node Θ_{k+1}.

The eventual estimate of Θ is given by $\hat{\theta}_L$. Note that the update (4.120) only involves the gradient of the local function f_k, whereas (4.119) requires the gradients of *all* local functions. The procedure of Fig. 4.32(b) is in the literature referred to as *stochastic approximation* (SA).

The stochastic approximation scheme can not only be applied to the estimation of *fixed* parameters, but also to filtering and smoothing in state-space models. The SA algorithm for estimating constant parameters can *directly* be used for *filtering* in state-space models. In the notation of Fig. 4.19 (left), the expression (4.120) takes the form:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \lambda_k \nabla_\theta \log f_B(\theta_k, y_k)\Big|_{\hat{\theta}_{k-1}}. \tag{4.121}$$

The SA algorithm for filtering may be viewed as forward-only message passing in the graph of the state-space model, as illustrated in Fig. 4.33. Note that update rule (4.121) takes the factor f_B into account, but it ignores the factor f_A. Accordingly, the message update at the node f_A is trivial: the output message is identical to the input message $\hat{\theta}_{k-1}$.

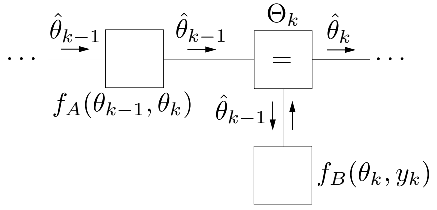


Figure 4.33: SA gradient method for filtering. Only one time slice of the state-space model is shown.

Along the same lines, SA gradient methods can be used for smoothing (see Fig. 4.34): one alternates a forward sweep of updates

$$\hat{\theta}_k^{\mathrm{F}} \triangleq \hat{\theta}_{k-1}^{\mathrm{F}} + \lambda \left. \frac{d \log f_B(\theta_k, y_k)}{d\theta_k} \right|_{\hat{\theta}_{k-1}^{\mathrm{F}}}, \tag{4.122}$$

with a backward sweep of updates

$$\hat{\theta}_k^{\mathrm{B}} \triangleq \hat{\theta}_{k+1}^{\mathrm{B}} - \lambda \left. \frac{d \log f_B(\theta_k, y_k)}{d\theta_k} \right|_{\hat{\theta}_{k+1}^{\mathrm{B}}}. \tag{4.123}$$

The forward and backward sweeps are initialized as $\hat{\theta}_1^{\mathrm{F}} \triangleq \hat{\theta}_1^{\mathrm{B}}$ and $\hat{\theta}_N^{\mathrm{B}} \triangleq \hat{\theta}_N^{\mathrm{F}}$ respectively. After a sufficient number of SA forward and backward sweeps, the eventual estimate $\hat{\theta}_k^{\mathrm{tot}}$ is obtained as the average of the forward and back estimate $\hat{\theta}_k^{\mathrm{F}}$ and $\hat{\theta}_k^{\mathrm{B}}$:

$$\hat{\theta}_k^{\mathrm{tot}} \triangleq \frac{1}{2}\big(\hat{\theta}_k^{\mathrm{F}} + \hat{\theta}_k^{\mathrm{B}}\big). \qquad (4.124)$$

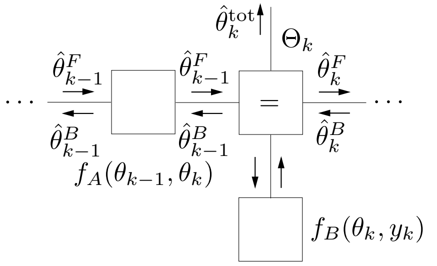


Figure 4.34: SA gradient method for smoothing.

Remark 4.8. (Stochastic approximation: pro and con)

- Pro:
  SA gradient algorithms can be implemented *without any knowledge of f_A*. Or, in other words, it is not necessary to model f_A explicitly.
- Con:
  Since SA gradient algorithms do not take the prior model f_A into account, they often have a *larger* estimation error than algorithms that do take f_A into account.

4.8.4 Application

Constant-Phase Model

We present two different sum-product-based gradient methods for the constant-phase model. The first algorithm is obtained by straightforwardly applying the generic rules of Section 4.8.1. The second method is an SA algorithm.

The first algorithm perform the following steps (see Fig. 4.35):

① The compound equality constraint node broadcasts the current phase estimate $\hat{\theta}^{\text{old}}$ to all multiply nodes.

② The latter reply with the messages $\left.\frac{d \log \mu_{g \to \Theta_k}(\theta)}{d\theta}\right|_{\theta=\hat{\theta}^{\text{old}}}$.

③ At the compound equality constraint node, a new phase estimate is computed according to the rule

$$\begin{aligned} \hat{\theta}^{\text{new}} &= \hat{\theta}^{\text{old}} + \lambda \left.\frac{d \log \mu_{\boxminus \to \Theta}(\theta)}{d\theta}\right|_{\theta=\hat{\theta}^{\text{old}}} && (4.125) \\ &= \hat{\theta}^{\text{old}} + \lambda \sum_{k=1}^{L} \left.\frac{d \log \mu_{g \to \Theta_k}(\theta)}{d\theta}\right|_{\theta=\hat{\theta}^{\text{old}}}. && (4.126) \end{aligned}$$

④ The upward messages $\mu_{\boxtimes \to X_k}(x_k)$ are computed:

$$\mu_{\boxtimes \to X_k}(x_k) \overset{\triangle}{\propto} \mu_{Z_k \to \boxtimes}\big(x_k \exp\big(j\hat{\theta}^{\text{new}}\big)\big). \qquad (4.127)$$

The steps ①–③ are iterated a number of times before ④ is performed. The initial estimate $\hat{\theta}$ may be obtained from a non-data aided algorithm (e.g., the M-law [136]).

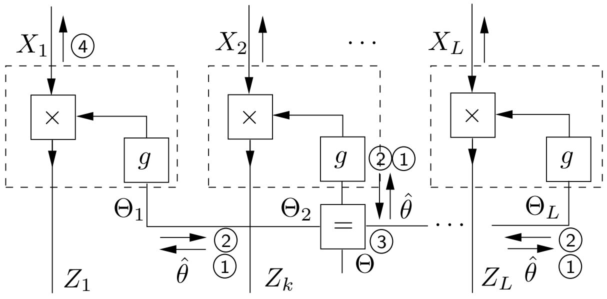


Figure 4.35: Gradient descent (of sum-product messages) in the factor graph of the constant-phase model: standard approach.

The derivatives in the RHS of (4.126) are computed as follows:

$$\begin{aligned}\frac{d\log\mu_{g\to\Theta_k}(\theta)}{d\theta} &= \frac{d\mu_{g\to\Theta_k}(\theta)}{d\theta}\mu^{-1}_{g\to\Theta_k}(\theta) &(4.128)\\ &= -\frac{1}{\sigma_N^2}\Big(\sum_{x_k}\mu_{X_k\to\boxtimes}(x_k)e^{-|x_k e^{j\theta_k}-y_k|^2/2\sigma_N^2}\\ &\quad\cdot\big[\sin\theta_k\mathrm{Re}(x_k y_k^*)+\cos\theta_k\mathrm{Im}(x_k y_k^*)\big]\Big)\\ &\quad\cdot\Big(\sum_{x_k}\mu_{X_k\to\boxtimes}(x_k)e^{-|x_k e^{j\theta_k}-y_k|^2/2\sigma_N^2}\Big)^{-1}, &(4.129)\end{aligned}$$

where y_k^* stands for the complex conjugate of y_k.

The SA scheme (depicted in Fig. 4.36) is a two-step procedure:

① The following operations are performed in a forward sweep ($k = 1, \ldots, N$):

a) The equality constraint node Θ_k sends the estimate $\hat{\theta}^{(k-1)}$ to the node f_k., which replies with the message $\nabla_\theta \log f_k(\theta)|_{\hat{\theta}^{(k-1)}}$.

b) A new estimate $\hat{\theta}^{(k)}$ is computed as

$$\hat{\theta}^{(k)} = \hat{\theta}^{(k-1)} + \lambda_k \left.\frac{d\log\mu_{g\to\Theta_k}(\theta)}{d\theta}\right|_{\hat{\theta}^{(k-1)}}, \tag{4.130}$$

which is then sent to the equality constraint node Θ_{k+1}.

The eventual estimate of Θ is given by $\hat{\theta}_L$.

② The upward messages $\mu_{\boxtimes\to X_k}(x_k)$ are computed:

$$\mu_{\boxtimes\to X_k}(x_k) \overset{\triangle}{\propto} \mu_{Z_k\to\boxtimes}\big(x_k\exp(j\hat{\theta}_L)\big). \tag{4.131}$$

Random-Walk Phase Model

Also for the random-walk phase model we present a “standard” gradient method and an SA gradient method.

In the first approach, one alternates the following steps:

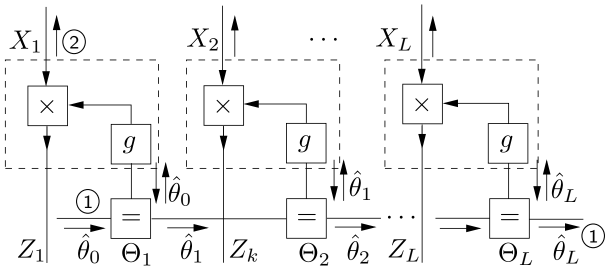


Figure 4.36: Gradient descent (of sum-product messages) in the factor graph of the constant-phase model: SA algorithm.

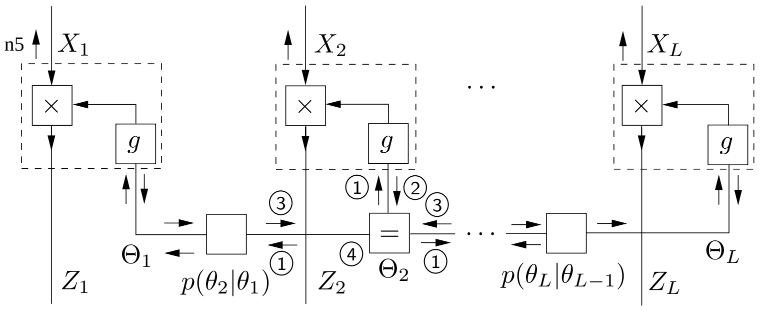


Figure 4.37: Gradient descent (of sum-product messages) in the factor graph of the random-walk phase model: standard approach.

① The equality constraint nodes broadcast the current phase estimates θ_k^{old} to the neighboring multiply nodes and phase noise nodes.

② The multiply nodes reply with the messages $\left.\frac{d \log \mu_{g\to\Theta_k}(\theta_k)}{d\theta}\right|_{\theta_k=\hat{\theta}_k^{\text{old}}}$.

③ The phase noise nodes reply with the messages $\left.\frac{d \log p(\theta_k|\theta_{k-1})}{d\theta_k}\right|_{\theta=\hat{\theta}^{\text{old}}}$ and $\left.\frac{d \log p(\theta_{k+1}|\theta_k)}{d\theta_k}\right|_{\theta=\hat{\theta}^{\text{old}}}$.

④ At the equality constraint nodes, a new phase estimate is computed according to the rule

$$\begin{aligned}\hat{\theta}^{\text{new}} &= \hat{\theta}^{\text{old}} + \lambda \left(\frac{d \log \mu_{g\to\Theta_k}(\theta_k)}{d\theta_k} \right. \\ &\qquad \left. + \frac{d \log p(\theta_k|\theta_{k-1})}{d\theta_k} + \frac{d \log p(\theta_{k+1}|\theta_k)}{d\theta_k} \right) \Bigg|_{\theta=\hat{\theta}^{\text{old}}}. \end{aligned} \qquad (4.132)$$

⑤ The upward messages $\mu_{\boxtimes\to X_k}(x_k)$ are computed:

$$\mu_{\boxtimes\to X_k}(x_k) \overset{\triangle}{\propto} \mu_{Z_k\to\boxtimes}\big(x_k \exp(j\hat{\theta}_k^{\text{new}})\big). \qquad (4.133)$$

The steps ①–④ are iterated a suitable number of times. The initial value of the phase Θ_k can be constant, i.e., $\hat{\theta}_k \triangleq \hat{\theta}$. It may also be generated by a forward SA sweep.

The derivative of $\log \mu_{g\to\Theta_k}(\theta_k)$ (cf. RHS of (4.132)) is computed as in (4.129). The derivatives of $\log p(\theta_k|\theta_{k-1})$ and $\log p(\theta_{k+1}|\theta_k)$ w.r.t. θ_k are computed as:

$$\begin{aligned}&\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_k} \\ &\qquad = -\frac{1}{\sigma_W^2}\frac{\sum_{n\in\mathbf{Z}}(\theta_k-\theta_{k-1}+n2\pi)e^{-(\theta_k-\theta_{k-1}+n2\pi)^2/2\sigma_W^2}}{\sum_{n\in\mathbf{Z}} e^{-(\theta_k-\theta_{k-1}+n2\pi)^2/2\sigma_W^2}},\end{aligned} \qquad (4.134)$$

and

$$\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_{k-1}} = -\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_k}. \qquad (4.135)$$

If σ_W is small, i.e., $\sigma_W \ll \pi$, then the evaluation of the RHS of (4.134) leads to numerical problems: for large values of the difference $\theta_k - \theta_{k-1}$, both the numerator and denominator in the RHS of (4.134) are zero. This problem can be circumvented by the approximation:

$$\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_k} \approx -\frac{1}{\sigma_W^2}\frac{\Delta_\theta + (\Delta_\theta + a)e^{-\frac{4\pi^2+b}{2\sigma_W^2}}}{1+e^{-\frac{4\pi^2+b}{2\sigma_W^2}}}, \qquad (4.136)$$

where $\sigma_W \ll \pi$, $\Delta_\theta \triangleq \theta_k - \theta_{k-1}$,

$$a \triangleq \begin{cases} 2\pi & \text{if } \Delta_\theta \leq 0 \\ -2\pi & \text{otherwise,} \end{cases} \qquad (4.137)$$

and

$$b \triangleq \begin{cases} 4\pi\Delta_\theta & \text{if } \Delta_\theta \leq 0 \\ -4\pi\Delta_\theta & \text{otherwise.} \end{cases} \tag{4.138}$$

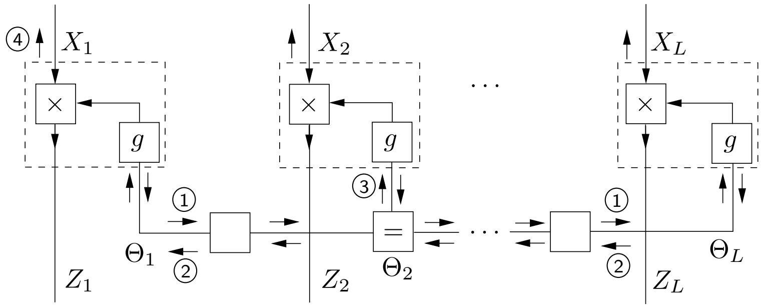


Figure 4.38: Gradient descent (of sum-product messages) in the factor graph of the random-walk phase model: SA algorithm.

In the SA algorithm (see Fig. 4.38), one alternates between a forward sweep of SA updates ①

$$\hat{\theta}_k^F = \hat{\theta}_{k-1}^F + \lambda \left. \frac{d \log \mu_{g\to\Theta_k}(\theta)}{d\theta} \right|_{\theta=\hat{\theta}_{k-1}^{FW}}. \tag{4.139}$$

and a backward sweep of SA updates ②

$$\hat{\theta}_k^B = \hat{\theta}_{k+1}^B - \lambda \left. \frac{d \log \mu_{g\to\Theta_k}(\theta)}{d\theta} \right|_{\theta=\hat{\theta}_{k+1}^{B}}. \tag{4.140}$$

The forward and backward sweeps are initialized as $\hat{\theta}_1^{\text{F}} \triangleq \hat{\theta}_1^{\text{B}}$ and $\hat{\theta}_N^{\text{B}} \triangleq \hat{\theta}_N^{\text{F}}$ respectively. After a sufficient number of forward and backward sweeps, the eventual estimate is computed as the average of the forward and backward estimate ③:

$$\hat{\theta}_k^{\text{tot}} \triangleq \frac{1}{2}\big(\hat{\theta}_k^{\text{F}} + \hat{\theta}_k^{\text{B}}\big). \tag{4.141}$$

The upward messages $\mu_{\boxtimes\to X_k}$ are computed as ④

$$\mu_{\boxtimes\to X_k}(x_k) \overset{\triangle}{\propto} \mu_{Z_k\to\boxtimes}\big(x_k \exp(j\hat{\theta}_k^{\text{tot}})\big). \tag{4.142}$$

4.8.5 Summary

We summarize the key points of this section.

- When steepest descent is combined with the sum-product algorithm, **gradients of sum-product messages** are required.
- If the local node function g is **differentiable**, the gradient of the outgoing message is computed by the sum-product rule applied to $\nabla_\theta g$, where the incoming messages are standard sum-product messages (see (4.101)). In other words, the differentiation operator does not propagate through the node g; it is only applied to the local node function g.
- If the local node g corresponds to a **deterministic mapping** h, the gradient of the outgoing message is computed by the sum-product rule applied to $\nabla_\theta h$ (see (4.115)). All incoming messages are standard sum-product messages, except for one, which is the *gradient* of an incoming sum-product message μ_Y. In this case, the differentiation operator is applied to both the local node function and the incoming message μ_Y; in other words, the differentiation operator propagates from node h towards the node the message μ_Y has been sent from.
- Differentiation also propagates through the **equality constraint node** (see (4.99) and (4.100)).
- The three previous observations indicate that along an edge in the factor graph, the following **messages** may propagate
  - standard sum-product messages,
  - gradients of sum-product messages,
  - hard decisions $\hat{\theta}$,

  depending on

  - the location of the edges at which the steepest descent update rules are applied
  - the kind of nodes that are involved.
- The sum-product messages and their gradients may be **represented** in various ways. In this fashion, steepest descent can readily and systematically be **combined** with other standard methods.

- We have interpreted Iterative Conditional Modes (ICM) and stochastic approximation as message passing:
  - In **ICM**, messages are represented by their **mode**.
  - **Stochastic approximation** can be viewed as a gradient method for parameter and state estimation with a specific message-update **schedule**.

4.9 Expectation Maximization

Expectation Maximization (EM) is a popular estimation algorithm. Both the EM algorithm and (iterative) sum-product algorithm are often viewed as alternative methods for estimating the parameters of graphical models. In other applications, the graphical model is used to compute the E-step of the EM algorithm [69]. In this section, we will show how the EM algorithm can be computed by local message updates on the factor graph. The results in this section are based on joint work Sascha Korl presented in [45] [100]. Earlier work on this topic was done by Eckford [61] independently. A related idea is proposed in [82].

4.9.1 General Idea

We begin by reviewing the expectation maximization (EM) algorithm. Suppose we wish to find

$$\hat{\theta}^{\max} \triangleq \underset{\theta}{\operatorname{argmax}}\, f(\theta). \tag{4.143}$$

We assume that $f(\theta)$ is the "marginal" of some real-valued function $f(x, \theta)$:

$$f(\theta) = \sum_x f(x, \theta), \tag{4.144}$$

where $\sum_x g(x)$ denotes either integration or summation of $g(x)$ over the whole range of x. The function $f(x, \theta)$ is assumed to be non-negative:

$$f(x, \theta) \geq 0 \quad \text{for all } x \text{ and all } \theta. \tag{4.145}$$

We will also assume that the integral (or the sum) $\sum_x f(x, \theta) \log f(x, \theta')$ exists for all θ, θ'. The described problem arises, for example, in the context of parameter estimation in state-space models. In such a context, the variable x is itself a vector and the function $f(x, \theta)$ has a non-trivial factor graph (see, for example, Fig. 4.40).

The EM algorithm attempts to compute (4.143) as follows:

a) Make some initial guess $\hat{\theta}^{(0)}$.

b) Expectation step: evaluate

$$f^{(k)}(\theta) \triangleq \sum_x f(x, \hat{\theta}^{(k)}) \log f(x, \theta). \tag{4.146}$$

c) Maximization step: compute

$$\hat{\theta}^{(k+1)} \triangleq \operatorname*{argmax}_\theta f^{(k)}(\theta). \tag{4.147}$$

d) Repeat 2–3 until convergence or until the available time is over.

The main property of the EM algorithm is

Theorem 4.1.

$$f(\hat{\theta}^{(k+1)}) \geq f(\hat{\theta}^{(k)}). \tag{4.148}$$

To prove this property, we need the following lemma.

Lemma 4.1. The function

$$\tilde{f}(\theta, \theta') \triangleq f(\theta') + \sum_x f(x, \theta') \log \frac{f(x, \theta)}{f(x, \theta')} \tag{4.149}$$

satisfies both

$$\tilde{f}(\theta, \theta') \leq f(\theta) \tag{4.150}$$

and

$$\tilde{f}(\theta, \theta) = f(\theta). \tag{4.151}$$

□

Proof: The equality (4.151) is obvious. The inequality (4.150) follows from eliminating the logarithm in (4.149) by the inequality $\log x \leq x - 1$ for $x > 0$:

$$\begin{aligned} \tilde{f}(\theta, \theta') &\leq f(\theta') + \sum_x f(x, \theta') \left(\frac{f(x,\theta)}{f(x,\theta')} - 1 \right) && (4.152) \\ &= f(\theta') + \sum_x f(x,\theta) - \sum_x f(x,\theta') && (4.153) \\ &= f(\theta). && (4.154) \end{aligned}$$

□

To prove (4.1), we first note that (4.147) is equivalent to

$$\hat{\theta}^{(k+1)} = \operatorname*{argmax}_{\theta} \tilde{f}(\theta, \hat{\theta}^{(k)}). \qquad (4.155)$$

We then obtain

$$\begin{aligned} f(\hat{\theta}^{(k)}) &= \tilde{f}(\hat{\theta}^{(k)}, \hat{\theta}^{(k)}) && (4.156) \\ &\leq \tilde{f}(\hat{\theta}^{(k+1)}, \hat{\theta}^{(k)}) && (4.157) \\ &\leq f(\hat{\theta}^{(k+1)}), && (4.158) \end{aligned}$$

where (4.156) follows from (4.151), (4.157) follows from (4.155), and (4.158) follows from (4.150).

Corollary 4.1. The global maximum $\theta^{\max}$ of $f(\theta)$ (cf. (4.143)) is a fixed point of EM.

Proof: Assume $\hat{\theta}^{(k)} = \theta^{\max}$. Since the EM algorithm never decreases f (cf. (4.1)), it follows that $\hat{\theta}^{(\ell)} = \theta^{\max}$, for all $\ell > k$, hence $\theta^{\max}$ is a fixed point. □

We prove two additional interesting properties.

Theorem 4.2. The fixed points of EM are stationary points of $f(\theta)$.

Note that this statement does not imply that *all* stationary points of $f(\theta)$ are fixed points of EM!

Proof: We will use the short-hand notation

$$\nabla_\theta g(\hat{\theta}) \triangleq \nabla_\theta g(\theta)|_{\theta=\hat{\theta}}\,. \tag{4.159}$$

The fixed points $\hat{\theta}^{\text{fixed}}$ of EM are implicitly defined as:

$$\hat{\theta}^{\text{fixed}} \overset{!}{=} \operatorname*{argmax}_{\theta} \sum_x f(x, \hat{\theta}^{\text{fixed}}) \log f(x, \theta). \tag{4.160}$$

If we define the function $\bar{f}(\theta, \theta')$ as

$$\bar{f}(\theta, \theta') \triangleq \sum_x f(x, \theta') \log f(x, \theta), \tag{4.161}$$

we can rewrite (4.160) as

$$\hat{\theta}^{\text{fixed}} \overset{!}{=} \operatorname*{argmax}_{\theta} \bar{f}(\theta, \theta^{\text{fixed}}). \tag{4.162}$$

At the fixed points, the first-order derivative of $\bar{f}(\theta, \theta')$ w.r.t. θ vanishes:

$$\nabla_\theta \bar{f}(\hat{\theta}^{\text{fixed}}, \hat{\theta}^{\text{fixed}}) \overset{!}{=} 0. \tag{4.163}$$

Note that

$$\begin{aligned}
\nabla_\theta \bar{f}(\theta, \theta') &\triangleq \nabla_\theta \sum_x f(x, \theta') \log f(x, \theta) && (4.164)\\
&= \sum_x f(x, \theta') \nabla_\theta \log f(x, \theta) && (4.165)\\
&= \sum_x f(x, \theta') \nabla_\theta f(x, \theta) / f(x, \theta). && (4.166)
\end{aligned}$$

In (4.165) we differentiated under the integral sign. We assume in this thesis that this operation is allowed.[8] From (4.166) it follows:

$$\begin{aligned}
\nabla_\theta \bar{f}(\hat{\theta}, \hat{\theta}) &= \sum_x \nabla_\theta f(x, \hat{\theta}) && (4.167)\\
&= \nabla_\theta f(\hat{\theta}), && (4.168)
\end{aligned}$$

for all $\hat{\theta}$. As a consequence of (4.163) and (4.168),

$$\nabla_\theta f(\hat{\theta}^{\text{fixed}}) = 0, \tag{4.169}$$

and hence the EM fixed points are stationary points (or "zero-gradient points") of $f(\theta)$. □

[8] In Appendix 6, we list necessary conditions for differentiation under the integral sign.

Theorem 4.3. A stationary point $\hat{\theta}^{\text{stat}}$ of f is a fixed point of EM, if $\bar{f}(\theta, \hat{\theta}^{\text{stat}})$ is concave in θ.

Proof: Stationary points $\hat{\theta}^{\text{stat}}$ of f are implicitly defined as:

$$\nabla_\theta f(\hat{\theta}^{\text{stat}}) \overset{!}{=} 0. \tag{4.170}$$

As a consequence of (4.168), it follows

$$\nabla_\theta \bar{f}(\hat{\theta}^{\text{stat}}, \hat{\theta}^{\text{stat}}) = 0. \tag{4.171}$$

Since by assumption $\bar{f}(\theta, \hat{\theta}^{\text{stat}})$ is concave in θ, we conclude that

$$\hat{\theta}^{\text{stat}} = \underset{\theta}{\operatorname{argmax}}\, \bar{f}(\theta, \hat{\theta}^{\text{stat}}), \tag{4.172}$$

which means that θ^{stat} is a fixed point of EM (cf. (4.350)). □

From Theorem 4.1 and 4.2 follows:

Corollary 4.2. If a local maximum of $f(\theta)$ is a fixed point of EM, then it is a *stable* fixed point. If a local minimum or saddle point of $f(\theta)$ is a fixed point of EM, then it is an *unstable* fixed point.

The proof is straightforward but a little technical; we omit it here.

Message-Passing Interpretation

We now rewrite the EM algorithm in message-passing form. In this section, we will assume a trivial factorization

$$f(x, \theta) = f_{\text{A}}(\theta) f_{\text{B}}(x, \theta), \tag{4.173}$$

where $f_{\text{A}}(\theta)$ may be viewed as encoding the a priori information about Θ. More interesting factorizations (i.e., models with internal structure) will be considered in the next section. The factor graph of (4.173) is shown in Fig. 4.39. In this setup, the EM algorithm amounts to iterative re-computation of the following messages:

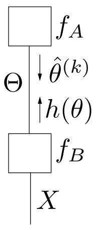


Figure 4.39: Factor graph of (4.173).

Upwards message $h(\theta)$:

$$h(\theta) = \frac{\sum_x f_{\mathrm{B}}(x, \hat{\theta}^{(k)}) \log f_{\mathrm{B}}(x, \theta)}{\sum_x f_{\mathrm{B}}(x, \hat{\theta}^{(k)})} \tag{4.174}$$

$$= \mathrm{E}_{p_{\mathrm{B}}}[\log f_{\mathrm{B}}(X, \theta)], \tag{4.175}$$

where $\mathrm{E}_{p_{\mathrm{B}}}$ denotes the expectation with respect to the probability distribution

$$p_{\mathrm{B}}(x|\hat{\theta}^{(k)}) \triangleq \frac{f_{\mathrm{B}}(x, \hat{\theta}^{(k)})}{\sum_{x'} f_{\mathrm{B}}(x', \hat{\theta}^{(k)})}. \tag{4.176}$$

Downwards message $\hat{\theta}^{(k)}$:

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \left(\log f_{\mathrm{A}}(\theta) + h(\theta)\right) \tag{4.177}$$

$$= \underset{\theta}{\operatorname{argmax}} \left(f_{\mathrm{A}}(\theta) \cdot e^{h(\theta)}\right). \tag{4.178}$$

The equivalence of this message-passing algorithm with (4.146) and (4.147) may be seen as follows. From (4.146) and (4.147), we have

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \sum_x f(x, \hat{\theta}^{(k)}) \log f(x, \theta) \tag{4.179}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_x f_{\mathrm{A}}(\hat{\theta}^{(k)}) f_{\mathrm{B}}(x, \hat{\theta}^{(k)}) \log\big(f_{\mathrm{A}}(\theta) f_{\mathrm{B}}(x, \theta)\big) \tag{4.180}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_x f_{\mathrm{B}}(x, \hat{\theta}^{(k)}) \Big(\log f_{\mathrm{A}}(\theta) + \log f_{\mathrm{B}}(x, \theta)\Big) \tag{4.181}$$

$$= \underset{\theta}{\operatorname{argmax}} \left(\log f_{\mathrm{A}}(\theta) + \frac{\sum_x f_{\mathrm{B}}(x, \hat{\theta}^{(k)}) \log f_{\mathrm{B}}(x, \theta)}{\sum_{x'} f_{\mathrm{B}}(x', \hat{\theta}^{(k)})}\right), \tag{4.182}$$

which is equivalent to (4.174) and (4.177).

Some remarks:

a) The computation (4.174) or (4.175) is *not* an instance of the sum-product algorithm.

b) The message $h(\theta)$ may be viewed as a "log-domain" summary of f_B. In (4.178), the corresponding "probability domain" summary $e^{h(\theta)}$ is consistent with the factor graph interpretation.

c) A constant may be added to (or subtracted from) $h(\theta)$ without affecting (4.177).

d) If $f_\mathrm{A}(\theta)$ is a constant, the normalization in (4.174) can be omitted. More generally, the normalization in (4.174) can be omitted if $f_\mathrm{A}(\theta)$ is constant for all θ such that $f_\mathrm{A}(\theta) \neq 0$. However, in contrast to most standard accounts of the EM algorithm, we explicitly wish to allow more general functions f_A.

e) Nothing changes if we introduce a known observation (i.e., a constant argument) y into f such that (4.173) becomes $f(x, y, \theta) = f_\mathrm{A}(y, \theta) f_\mathrm{B}(x, y, \theta)$.

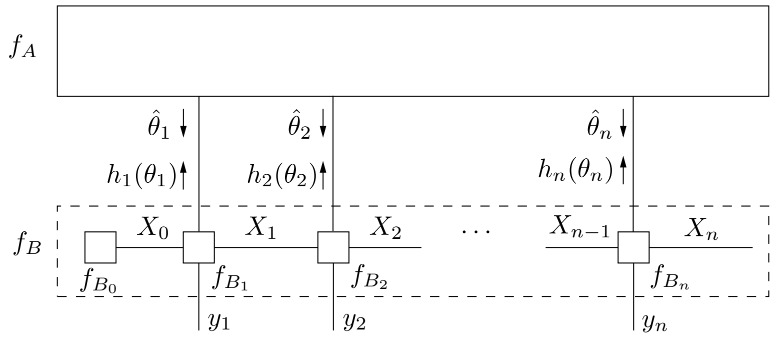


Figure 4.40: Factor graph of (4.214).

A Simple Example

Suppose that a binary symbol X (with values in $\{+1, -1\}$) is transmitted over an additive white Gaussian noise (AWGN) channel with an unknown real gain Θ. The channel output is

$$Y = \Theta \cdot X + W \tag{4.183}$$

where W is a zero-mean Gaussian random variable (independent of X and Θ) with known variance σ^2. Based on the (fixed) observation $Y = y$, we wish to estimate Θ. (We also note an inherent ambiguity in the problem: the sign of the gain Θ and the sign of X cannot be determined individually.)

A factor graph of this system model is shown in Fig. 4.41. The node symbols in this factor graph and the corresponding factors are listed in Table 4.1. The small dashed box labeled "p_Θ" represents an a priori distribution over Θ, which may or may not be available.

If such a prior p_Θ is available, the factor graph represents the probability distribution

$$p(x, \theta \mid y) \propto p(y, \theta \mid x) \tag{4.184}$$

(when the other variables are eliminated by marginalization); when no such prior is available, the factor graph represents

$$p(x \mid \theta, y) \propto p(y \mid x, \theta). \tag{4.185}$$

The factor graph of Fig. 4.41 has no cycles. Therefore, exact marginals can be computed by the sum-product algorithm (without iterations). Although our interest is in the EM-algorithm, it is instructive to begin by writing down the messages of the sum-product algorithm. The sum-product message towards the left along the edge labeled Z ($\triangleq \Theta \cdot X$) is

$$\overleftarrow{\mu}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-z)^2}{2\sigma^2} \right) \tag{4.186}$$

and the sum-product message upwards along the edge Θ is

$$\mu{\uparrow}(\theta) \;=\; \sum_x \sum_{-\infty}^{\infty} \delta(z - x \cdot \theta)\, \overleftarrow{\mu}(z)\, dz \tag{4.187}$$

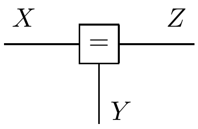
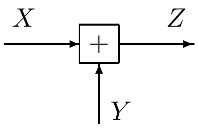
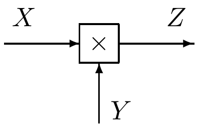
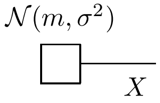
symbol	factor
	$\delta(z-x)\delta(z-y)$
	$\delta(z-x+y)$
	$\delta(z-x\cdot y)$
	$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$

Table 4.1: Some node symbols and the corresponding local functions (factors).

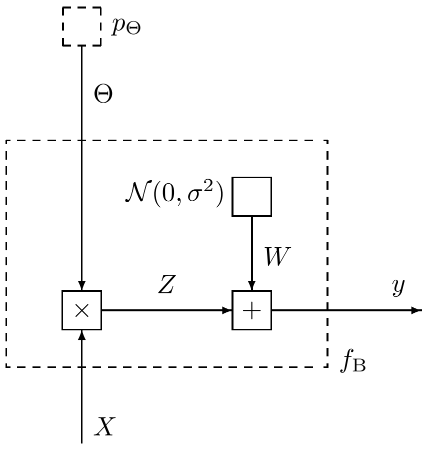


Figure 4.41: Factor graph corresponding to (4.183).

$$\propto \sum_x \exp\left(-\frac{(y-x\cdot\theta)^2}{2\sigma^2}\right) \tag{4.188}$$

$$= \exp\left(-\frac{(y-\theta)^2}{2\sigma^2}\right) + \exp\left(-\frac{(y+\theta)^2}{2\sigma^2}\right). \tag{4.189}$$

If a prior p_Θ is given, the a posteriori distribution of Θ is

$$p(\theta|y) \propto p_\Theta(\theta)\,\mu{\uparrow}(\theta). \tag{4.190}$$

When no such prior is available, the a posteriori distribution of Θ is

$$p(\theta|y) \propto \mu{\uparrow}(\theta). \tag{4.191}$$

Fig. 4.42 shows the message $\mu{\uparrow}(\theta)$ for several values of σ and a fixed value of y, i.e., $y \triangleq 1$. Fig. 4.43 shows how the mode $\theta^{\max}$ of the message $\mu{\uparrow}(\theta)$ depends on σ^2.

We now apply the EM algorithm to this example. The big dashed box in Fig. 4.41 will take the role of f_B in Fig. 4.39. If we close this box by

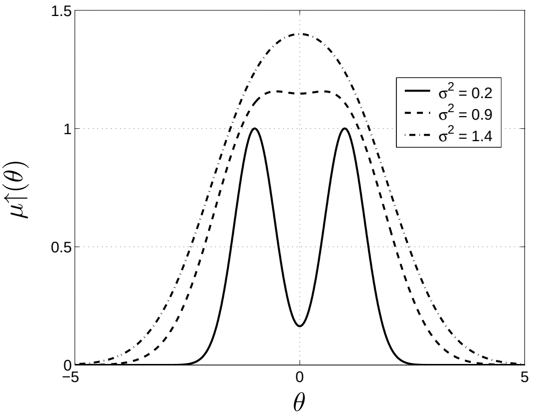


Figure 4.42: The message $\mu\!\uparrow\!(\theta)$ for several values of σ with $y = 1$.

marginalizing out its internal variables W and Z, we obtain

$$f_{\mathrm{B}}(x, y, \theta) \;=\; \sum_{-\infty}^{\infty} \delta(z - x \cdot \theta)\, \overleftarrow{\mu}(z)\, dz \tag{4.192}$$

$$\;=\; \frac{1}{\sqrt{2\pi\sigma^2}} \exp\!\left(-\frac{(y - x\cdot\theta)^2}{2\sigma^2}\right). \tag{4.193}$$

The E-log message (4.174) is

$$h(\theta) = \gamma^{-1} \sum_x f_{\mathrm{B}}(x, y, \hat{\theta}^{(k)}) \log f_{\mathrm{B}}(x, y, \theta) \tag{4.194}$$

with

$$\gamma \triangleq \sum_x f_{\mathrm{B}}(x, y, \hat{\theta}^{(k)}). \tag{4.195}$$

With $\alpha^{(k)} \triangleq f_{\mathrm{B}}(+1, y, \hat{\theta}^{(k)})$ and $\beta^{(k)} \triangleq f_{\mathrm{B}}(-1, y, \hat{\theta}^{(k)})$, we have

$$\gamma = \alpha^{(k)} + \beta^{(k)} \tag{4.196}$$

and

$$h(\theta) \;=\; \gamma^{-1} \sum_x f_{\mathrm{B}}(x, y, \hat{\theta}^{(k)}) \left(-\log\!\left(\sqrt{2\pi\sigma^2}\right) - \frac{(y - x\cdot\theta)^2}{2\sigma^2}\right) \tag{4.197}$$

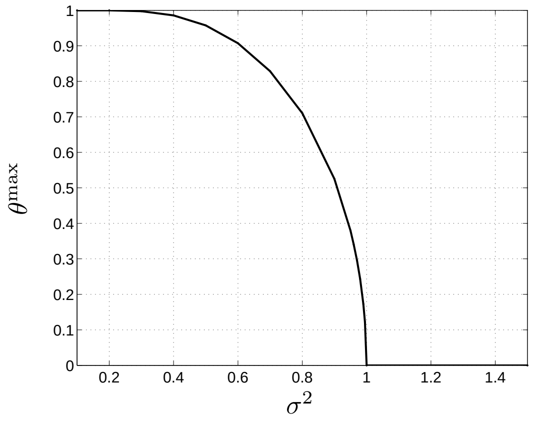


Figure 4.43: The (positive) mode $\theta^{\max}$ of the message $\mu\!\uparrow\!(\theta)$ as a function of σ^2 (with $y = 1$).

$$= \; -\log\Big(\sqrt{2\pi\sigma^2}\Big) - \gamma^{-1}\sum_x f_{\mathrm{B}}(x, y, \hat{\theta}^{(k)})\,\frac{(y - x\cdot\theta)^2}{2\sigma^2} \tag{4.198}$$

$$= \; -\log\Big(\sqrt{2\pi\sigma^2}\Big) - \frac{\alpha^{(k)}(y-\theta)^2 + \beta^{(k)}(y+\theta)^2}{2\sigma^2(\alpha^{(k)}+\beta^{(k)})} \tag{4.199}$$

The E-log message $h(\theta)$ is thus a quadratic form and $e^{h(\theta)}$ is a (unnormalized) Gaussian density; the mean and the variance of this Gaussian density are easily identified as

$$\text{mean of (normalized) } e^{h(\theta)} = \frac{\alpha^{(k)} - \beta^{(k)}}{\alpha^{(k)} + \beta^{(k)}}\, y \tag{4.200}$$

and

$$\text{variance of (normalized) } e^{h(\theta)} = \sigma^2. \tag{4.201}$$

Note that this is nicer than the sum-product message $\mu\!\uparrow\!(\theta)$ (4.189), which is a Gaussian mixture.

The downwards message (i.e., the next estimate) $\hat{\theta}^{(k+1)}$ is

$$\hat{\theta}^{(k+1)} = \operatorname*{argmax}_{\theta}\Big(p_\Theta(\theta)\cdot e^{h(\theta)}\Big) \tag{4.202}$$

according to (4.178). In the absence of a prior $p_\Theta(\theta)$, this simplifies to

$$\begin{aligned}
\hat{\theta}^{(k+1)} &= \operatorname*{argmax}_{\theta} e^{h(\theta)} && (4.203)\\
&= \frac{\alpha^{(k)} - \beta^{(k)}}{\alpha^{(k)} + \beta^{(k)}}\, y && (4.204)\\
&= y \cdot \frac{\exp\Big(-\frac{(y-\hat{\theta}^{(k)})^2}{2\sigma^2}\Big) - \exp\Big(-\frac{(y+\hat{\theta}^{(k)})^2}{2\sigma^2}\Big)}{\exp\Big(-\frac{(y-\hat{\theta}^{(k)})^2}{2\sigma^2}\Big) + \exp\Big(-\frac{(y+\hat{\theta}^{(k)})^2}{2\sigma^2}\Big)} && (4.205)\\
&= y \cdot \frac{e^{\hat{\theta}^{(k)}y/\sigma^2} - e^{-\hat{\theta}^{(k)}y/\sigma^2}}{e^{\hat{\theta}^{(k)}y/\sigma^2} + e^{-\hat{\theta}^{(k)}y/\sigma^2}} && (4.206)\\
&= y \cdot \tanh(\hat{\theta}^{(k)}y/\sigma^2), && (4.207)
\end{aligned}$$

where the step to (4.204) follows from (4.200).

Note that $\hat{\theta}^{(k+1)}$ has the same sign as $\hat{\theta}^{(k)}$ (independent of the sign of y); the inherent ambiguity of the sign of Θ is thus resolved by the choice of the starting value $\hat{\theta}^{(0)}$.

The fixed points of the recursion (4.207) are stationary points of $p(\theta|y) \propto \mu{\uparrow}(\theta)$. This follows from Theorem 4.3, since $h(\theta)$ (4.199) is concave. It can also directly be verified:

$$\begin{aligned}
\frac{d\mu{\uparrow}(\theta)}{d\theta} &= \frac{d}{d\theta}e^{-\frac{(y-\theta)^2}{2\sigma^2}} + \frac{d}{d\theta}e^{-\frac{(y+\theta)^2}{2\sigma^2}} && (4.208)\\
&= \frac{1}{\sigma^2}\Big((y-\theta)\cdot e^{-\frac{(y-\theta)^2}{2\sigma^2}} - (y+\theta)\cdot e^{-\frac{(y+\theta)^2}{2\sigma^2}}\Big) && (4.209)\\
&\overset{!}{=} 0 && (4.210)\\
&\Leftrightarrow && (4.211)\\
\theta &= y \cdot \frac{e^{\theta y/\sigma^2} - e^{-\theta y/\sigma^2}}{e^{\theta y/\sigma^2} + e^{-\theta y/\sigma^2}} && (4.212)\\
&= y \cdot \tanh(\theta y/\sigma^2). && (4.213)
\end{aligned}$$

So far, all we have done is to write a simple standard example of the EM algorithm in message-passing form. For this simple example, there is probably no advantage for doing so. The benefits of the message passing approach come with more richly structured models, which we will consider in the next section.

Nevertheless, the following aspects of this example carry over to many larger examples. Fist, we note that the E-log message is nicer than the sum-product message (Gaussian instead of a Gaussian mixture). Second, in the computation of the E-log message $h(\theta)$, we used the "internal" sum-product message $\overleftarrow{\mu}(z)$ in step (4.192);[9] moreover, (4.192) is itself a sum-product computation. We shall see that the use of the sum-product algorithm to compute E-log messages is a cornerstone of the message passing view of the EM algorithm.

Non-trivial Factor Graphs

The algorithm of the previous section still applies if $\Theta = (\Theta_1, \ldots, \Theta_n)^T$ and $X = (X_0, \ldots, X_n)^T$ are vectors. However, opportunities to simplify the computations may arise if f_A and f_B have "nice" factorizations. For example, assume that f_B factors as

$$f_\text{B}(x, y, \theta) = f_{B_0}(x_0) f_{B_1}(x_0, x_1, y_1, \theta_1) \cdots f_{B_n}(x_{n-1}, x_n, y_n, \theta_n), \tag{4.214}$$

where $y = (y_1, \ldots, y_n)^T$ is some known (observed) vector. Such factorizations arise from classical trellis models and state-space models. The factor graph corresponding to (4.214) is shown in Fig. 4.40.

The upwards message $h(\theta)$ (4.175) splits into a sum with one term for each node in the factor graph:

$$\begin{aligned} h(\theta) &= \text{E}\Big[\log\Big(f_{B_0}(x_0) f_{B_1}(x_0, x_1, y_1, \theta_1) \cdots \\ &\qquad \cdots f_{B_n}(x_{n-1}, x_n, y_n, \theta_n)\Big)\Big] && (4.215) \\ &= \text{E}[\log f_{B_0}(X_0)] + \text{E}[\log f_{B_1}(X_0, X_1, y_1, \theta_1)] + \ldots \\ &\qquad \ldots + \text{E}[\log f_{B_n}(X_{n-1}, X_n, y_n, \theta_n)] && (4.216) \end{aligned}$$

Each term

$$h_k(\theta_k) \stackrel{\triangle}{=} \text{E}[\log f_{B_k}(X_{k-1}, X_k, y_k, \theta_k)] \tag{4.217}$$

may be viewed as the message out of the corresponding node, as indicated in Fig. 4.40. The constant term $\text{E}[\log f_{B_0}(X_0)]$ in (4.216) may be omitted (cf. Remark c in Section 4.9.1). As in (4.175), all expectations are with respect to the probability distribution p_B, which we here denote by

[9]We come back to this issue in Section 4.9.3.

$p_{\mathrm{B}}(x|y,\hat{\theta})$. Note that each term (4.217) requires only $p_{\mathrm{B}}(x_{k-1},x_k|y,\hat{\theta})$, the joint distribution of X_{k-1} and X_k:

$$h_k(\theta_k) = \sum_{x_{k-1}} \sum_{x_k} p_{\mathrm{B}}(x_{k-1},x_k|y,\hat{\theta}) \log f_{B_k}(x_{k-1},x_k,y_k,\theta_k). \tag{4.218}$$

These joint distributions may be obtained by means of the standard sum-product algorithm: from elementary factor graph theory, we have

$$\begin{aligned} p_{\mathrm{B}}(x_{k-1},x_k|y,\hat{\theta}) \;\propto\; & f_{B_k}(x_{k-1},x_k,y_k,\hat{\theta})\, \mu_{X_{k-1}\to f_{B_k}}(x_{k-1}) \\ & \cdot \mu_{X_k\to f_{B_k}}(x_k), \end{aligned} \tag{4.219}$$

where $\mu_{X_{k-1}\to f_{B_k}}$ and $\mu_{X_k\to f_{B_k}}$ are the messages of the sum-product algorithm towards the node f_{B_k} and where "$\propto$" denotes equality up to a scale factor that does not depend on x_{k-1}, x_k. It follows that

$$\begin{aligned} & p_{\mathrm{B}}(x_{k-1},x_k|y,\hat{\theta}) \;=\; \\ & \frac{f_{B_k}(x_{k-1},x_k,y_k,\hat{\theta})\, \mu_{X_{k-1}\to f_{B_k}}(x_{k-1})\, \mu_{X_k\to f_{B_k}}(x_k)}{\sum_{x_{k-1}}\sum_{x_k} f_{B_k}(x_{k-1},x_k,y,\hat{\theta})\, \mu_{X_{k-1}\to f_{B_k}}(x_{k-1})\, \mu_{X_k\to f_{B_k}}(x_k)}. \end{aligned} \tag{4.220}$$

Note that, if the sum product messages $\mu_{X_{k-1}\to f_{B_k}}$ and $\mu_{X_k\to f_{B_k}}$ are computed without any scaling, then the denominator in (4.220) equals $p_{\mathrm{B}}(y|\hat{\theta})$, which is independent of k.

We now investigate two examples.

- The variables X may be **symbols** protected by a **trellis code**. If the state space of the code is not too large, the messages $\mu_{X_{k-1}\to f_{B_k}}$ and $\mu_{X_k\to f_{B_k}}$ may be computed by the forward and backward recursion of the BCJR algorithm [15] through the trellis of the code, or equivalently, by a forward and backward sum-product sweep on the factor graph of the code.

- The graph $f_B(x,\theta)$ may represent a **linear dynamical system** (see Appendix H and [100]). The messages $\mu_{X_{k-1}\to f_{B_k}}$ and $\mu_{X_k\to f_{B_k}}$ are then computed by a forward and backward Kalman recursion.

In some applications, the messages $\mu_{X_{k-1}\to f_{B_k}}$ and $\mu_{X_k\to f_{B_k}}$ are not available in closed-form; they may then be represented in various ways such as quantized messages or lists of samples; they may also be approximated by Gaussian distributions [100]. We come back to this issue in Section 4.9.5.

The downwards message $\hat{\theta}$ (4.177) is

$$(\hat{\theta}_1, \ldots, \hat{\theta}_n)^T = \underset{\theta_1,\ldots,\theta_n}{\operatorname{argmax}} \big(\log f_{\mathrm{A}}(\theta) + h_1(\theta_1) + h_n(\theta_n) \big) \tag{4.221}$$

$$= \underset{\theta_1,\ldots,\theta_n}{\operatorname{argmax}} \Big(f_{\mathrm{A}}(\theta) \cdot e^{h_1(\theta_1)} \cdots e^{h_n(\theta_n)} \Big). \tag{4.222}$$

If f_{A} has itself a nice factorization, then (4.221) or (4.222) may be computed by the standard max-sum or max-product algorithm respectively. This applies in particular for the standard case $\Theta_1 = \Theta_2 = \ldots = \Theta_n$ (see Fig. 4.44):

$$(\hat{\theta}_1, \ldots, \hat{\theta}_n)^T = \underset{\theta_1,\ldots,\theta_n}{\operatorname{argmax}} \big(h_1(\theta_1) + \cdots + h_n(\theta_n) \big) \tag{4.223}$$

$$= \underset{\theta_1,\ldots,\theta_n}{\operatorname{argmax}} \Big(e^{h_1(\theta_1)} \cdots e^{h_n(\theta_n)} \Big). \tag{4.224}$$

In addition, if the term $f_A(\theta)$ and all terms $e^{h_\ell(\theta_\ell)}$ are Gaussians, the max-product and the sum-product scheme are equivalent [117] (see Appendix H). Therefore, the Kalman filter can be used to solve (4.222), not to be confused with the Kalman filter of the E-step (4.216)–(4.218). We can use Gaussian sum-product message passing and the standard update rules given in Table H.2 and H.3 for this purpose.

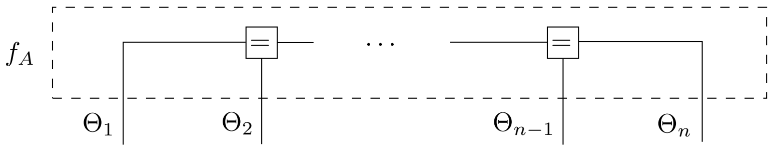


Figure 4.44: Factor graph of $\Theta_1 = \Theta_2 = \ldots = \Theta_n$.

The above derivations do not in any essential way depend on the specific example (4.214). In principle, any cut-set of edges in some factor graph may be chosen to be the vector Θ. However, the resulting subgraphs corresponding to f_{A} and f_{B} should be cycle-free in order to permit the computation of exact expectations (h-messages) and maximizations ($\hat{\theta}$-messages). The h-message out of a generic node $g(z_1, \ldots, z_m, \theta_k)$ (cf. Fig. 4.45) is as follows.

E-log message out of a generic node:

$$\begin{aligned} h(\theta_k) &= \sum_{z_1} \ldots \sum_{z_m} p(z_1, \ldots, z_m | \hat{\theta}_k) \log g(z_1, \ldots, z_m, \theta_k) && (4.225) \\ &= \gamma^{-1} \sum_{z_1} \ldots \sum_{z_m} g(z_1, \ldots, z_m, \hat{\theta}_k)\, \mu(z_1) \cdots \mu(z_m) \\ &\qquad \cdot \log g(z_1, \ldots, z_m, \theta_k) && (4.226) \end{aligned}$$

with

$$\gamma \stackrel{\triangle}{=} \sum_{z_1} \ldots \sum_{z_m} g(z_1, \ldots, z_m, \hat{\theta}_k)\, \mu(z_1) \cdots \mu(z_m) \qquad (4.227)$$

and where $\mu(z_1), \ldots, \mu(z_m)$ are standard sum-product messages.

If exact sum-product message passing (on a cycle-free graph) is intractable, the message passing rule (4.226) may also be applied to a (sub) graph with cycles, but then there is no guarantee for (4.1). Alternatively, one may use:

- **low-complexity approximations** such as (structured) mean-field approximations [94] [217] and extensions (see, e.g., [90]),
- **intermediate-complexity approximations** as for example structured-summary propagation [50] [51] or generalized belief propagation [223].

Also then, there is no guarantee for (4.1).

Remark 4.9. (Scheduling)
At every iteration of the (standard) EM algorithm (4.174)–(4.175), the messages are computed in a specific order ("standard schedule"):

a) With the current estimate $\hat{\theta}^{(k)}$, all sum-product messages $\mu(z_\ell)$ (cf. (4.226)) in the subgraph $f_B(x, \theta)$ are updated.

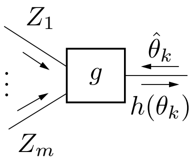


Figure 4.45: h-message out of a generic node.

b) The E-log messages are computed according to the rule (4.226) using the messages $\mu(z_\ell)$ updated in Step 1.

c) A new estimate $\hat{\theta}^{(k+1)}$ is computed according to (4.175).

One advantage of representing EM as message passing is the freedom in choosing a schedule that differs from the standard schedule.[10] For example, if the messages $\mu(z_\ell)$ are expensive to compute, it is natural to re-use the messages $\mu(z_\ell)$ from a previous iteration; in that case, one does *not* recompute the messages $\mu(z_\ell)$ at every iteration. This may lead to a substantial reduction of the computational complexity (see, e.g., [82]). More precisely, the number of flops needed to attain a fixed point of the EM algorithm can be significantly smaller. For the sake of clarity, we explicitly write down this alternative message-update schedule:

a) Initialize an estimate $\hat{\theta}^{(0)}$.

b) Iterate the following steps:

   i) Compute the messages $\mu(z_\ell)$.

   ii) Compute the messages $h(\theta)$; the messages $\mu(z_\ell)$ from (a) are plugged into (4.226).

   iii) Compute the new estimate $\hat{\theta}^{(k)}$ according to (4.177).

   iv) Iterate (b)–(c) until convergence.

[10]Note, however, that the choice of the update schedule does not change the fixed points of a message-passing algorithm, i.e., the set of solutions to which the algorithm may converge. However, the schedule determines *whether* the algorithm converges. In case the algorithm converges, scheduling determines to *which* fixed point the algorithm actually converges.

This is a double-loop algorithm; in the inner loop (b)–(c), the estimate $\hat{\theta}$ is refined, while the sum-product messages $\mu(z_\ell)$ are kept fixed. Obviously, one may fix *some* (i.e., not *all*) of the messages $\mu(z_\ell)$, and recompute the others at every iteration, depending on the complexity of updating $\mu(z_\ell)$.

Remark 4.10. (Deterministic nodes)
We consider now a node g that represents a deterministic mapping, i.e., $g(x, z, \theta) = \delta(z - f(x, \theta))$ (see Fig. 4.46). Literal application of (4.226) leads to

$$\begin{aligned} h(\theta) \;&=\; \gamma^{-1} \sum_x \sum_z \delta(z - f(x, \hat{\theta}^{(k)}))\, \mu_X(x) \mu_Z(z) \\ &\qquad\qquad \cdot \log \delta(z - f(x, \theta)). \end{aligned} \tag{4.228}$$

For simplicity, we assume that X and Z are discrete, and therefore

$$\begin{aligned} h(\theta) \;&=\; \gamma^{-1} \sum_x \sum_z \delta[z - f(x, \hat{\theta}^{(k)}))]\, \mu_X(x) \mu_Z(z) \log \delta[z - f(x, \theta)] & (4.229) \\ &=\; \gamma^{-1} \sum_x \mu_X(x) \mu_Z(f(x, \hat{\theta}^{(k)})) \log \delta[f(x, \hat{\theta}^{(k)}) - f(x, \theta)] & (4.230) \\ &=\; \begin{cases} -\infty & \text{if } f(x, \theta) \neq f(x, \theta'), \\ 0 & \text{otherwise.} \end{cases} & (4.231) \end{aligned}$$

We used the convention $0 \log 0 \stackrel{\triangle}{=} 0$ in (4.230). Assuming that

$$f(x, \theta) = f(x, \theta') \Leftrightarrow \theta = \theta', \forall x, \tag{4.232}$$

we can rewrite (4.231) as

$$h(\theta) \;=\; \begin{cases} -\infty & \text{if } \theta \neq \theta', \\ 0 & \text{otherwise,} \end{cases} \tag{4.233}$$

and as a consequence:

$$e^{h(\theta)} \;=\; \begin{cases} 0 & \text{if } \theta \neq \theta', \\ 1 & \text{otherwise.} \end{cases} \tag{4.234}$$

The solution (4.234) is disturbing: the message $h(\theta)$ has its maximum at the current estimate $\hat{\theta}^{(k)}$, and therefore, the estimate for θ is not refined in the course of the EM algorithm.

In conclusion: the E-log update rule leads to absurd results if it is applied to deterministic nodes. In Section 4.9.3, we propose a solution to this problem.

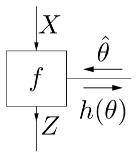


Figure 4.46: h-message out of a node $g(x, z, \theta) = \delta(z - f(x, \theta))$.

4.9.2 Table of EM update rules

In this section, the local E-log update rule (4.226) is worked out for a number of common nodes. All h-messages (or e^h) are collected in Table 4.2, the corresponding derivations can be found in Appendix J. The update rules in Table 4.2 can be used as building blocks for novel EM-based estimation algorithms, as demonstrated in [100]. In Table 4.2 and in Appendix J, we denote vectors in bold in order to avoid confusion between the *scalar* case and the *vector* case of message-update rules. So far, we did *not* denote vectors in bold in order to keep the notation as light as possible.

Table 4.2: EM Update equations for standard building blocks.

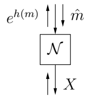
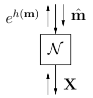
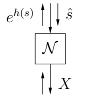
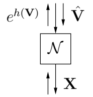
	Graph	Node	EM update rule	
1	 $f(x,m)=\mathcal{N}(x \mid m,s)$	Gaussian, unknown mean, scalar	$e^{h(m)} \propto \mathcal{N}\Big(m \,\Big|\, \mathrm{E}[X], s\Big)$	$X, m \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
2	 $e^{h(\mathbf{m})}$ $\hat{\mathbf{m}}$ $\mathcal{N}$ $\mathbf{X}$ $f(x,\mathbf{m})=\mathcal{N}(x \mid \mathbf{m},\mathbf{V})$	Gaussian, unknown mean	$e^{h(\mathbf{m})} \propto \mathcal{N}\Big(\mathbf{m} \,\Big|\, \mathrm{E}[\mathbf{X}], \mathbf{V}\Big)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} \in \mathbb{R}^{n\times n}$ $\mathbf{V} \succeq 0$
3	 $f(x,s)=\mathcal{N}(x \mid m,s)$	Gaussian, unknown variance	$e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{\mathrm{E}[X^2]-2m\mathrm{E}[X]+m^2}{2}\right)$	$X, m \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
4	 $e^{h(\mathbf{V})}$ $\hat{\mathbf{V}}$ $\mathcal{N}$ $\mathbf{X}$ $f(\mathbf{x},\mathbf{V})=\mathcal{N}(\mathbf{x} \mid \mathbf{m},\mathbf{V})$	Gaussian, unknown variance	$e^{h(\mathbf{V})} \propto \exp\left(-\mathrm{E}\left[(\mathbf{X}-\mathbf{m})^H\mathbf{V}^{-1}(\mathbf{X}-\mathbf{m})\right]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} \in \mathbb{R}^{n\times n}$ $\mathbf{V} \succeq 0$
5		Identity covariance matrix	$e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, \frac{n-2}{2}, \frac{1}{2}\mathrm{E}\left[(\mathbf{X}-\mathbf{m})^H(\mathbf{X}-\mathbf{m})\right]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} = \mathbf{I}s$ $s \in \mathbb{R}_0^+$

Table 4.2: (continued)

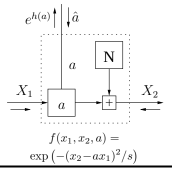
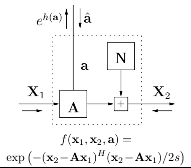
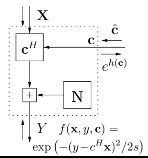
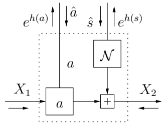
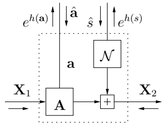
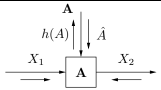
	Graph	Node	EM update rule	
6		Diagonal covariance matrix	$e^{h(s)} \propto \prod_{\ell=1}^{n} \mathrm{Ig}\left(s_\ell \,\middle\vert\, -\frac{1}{2}, \frac{1}{2}\mathrm{E}[(X_\ell - m_\ell)^2]\right)$	$\mathbf{X}, \mathbf{m} \in \mathbb{R}^n$ $\mathbf{V} = \mathrm{diag}\,(\mathbf{s})$ $\mathbf{s} \in \mathbb{R}^{+n}$
7	 $e^{h(a)}$ $\hat{a}$ a N X_1 X_2 a + $f(x_1, x_2, a) =$ $\exp\left(-(x_2 - a x_1)^2/s\right)$	Scalar multi-plication	$e^{h(a)} \propto \mathcal{N}^{-1}\left(a \,\middle\vert\, \frac{\mathrm{E}[X_1 X_2]}{\mathrm{E}[X_1^2]}, \frac{\mathrm{E}[X_1^2]}{s}\right)$	$X_1, X_2 \in \mathbb{R}$ $a \in \mathbb{R}$ $s \in \mathbb{R}_0^+$
8	 $e^{h(\mathbf{a})}$ $\hat{\mathbf{a}}$ $\mathbf{a}$ N $\mathbf{X}_1$ $\mathbf{X}_2$ A + $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}) =$ $\exp\left(-(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^H(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)/2s\right)$	Auto-regression	$e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\left(\mathbf{a} \,\middle\vert\, \mathrm{E}[\mathbf{X}_1 \mathbf{X}_1^H]^{-1} \mathrm{E}[\mathbf{X}_1 X_2], \frac{\mathrm{E}[\mathbf{X}_1 \mathbf{X}_1^H]}{s}\right)$	$\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^n$ $\mathbf{a} \in \mathbb{R}^n, s \in \mathbb{R}_0^+$ $\mathbf{A} = [\mathbf{a}^H\,;\, \mathbf{I}\ \mathbf{0}]$ $X_2 = [\mathbf{X}_2]_1$
9	 $\mathbf{X}$ $\hat{\mathbf{c}}$ $\mathbf{c}^H$ $\mathbf{c}$ $e^{h(\mathbf{c})}$ + N Y $f(\mathbf{x}, y, \mathbf{c}) =$ $\exp\left(-(y - c^H \mathbf{x})^2/2s\right)$	Inner vector product	$e^{h(\mathbf{c})} \propto \mathcal{N}^{-1}\left(\mathbf{c} \,\middle\vert\, \mathrm{E}[\mathbf{X}\mathbf{X}^H]^{-1} \mathrm{E}[\mathbf{X} Y], \frac{\mathrm{E}[\mathbf{X}\mathbf{X}^H]}{s}\right)$	$\mathbf{X}, \mathbf{c} \in \mathbb{R}^n$ $Y \in \mathbb{R}$ $s \in \mathbb{R}_0^+$

Table 4.2: (continued)

	Graph	Node	EM update rule	
10	 $e^{h(a)}$ $\hat{a}$ $\hat{s}$ $e^{h(s)}$ $\mathcal{N}$ a X_1 a $+$ X_2 $f(x_1,x_2,a,s) = \exp\left(-(x_2-ax_1)^2/2s\right)$	Joint coefficient/variance estimation scalar	$e^{h(a)} \propto \mathcal{N}^{-1}\left(a \,\middle\vert\, \frac{\mathrm{E}[X_1X_2]}{\mathrm{E}[X_1^2]}, \frac{\mathrm{E}[X_1^2]}{\hat{s}}\right)$ $e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle\vert\, -\frac{1}{2}, \frac{\mathrm{E}[(X_2-\hat{a}X_1)^2]}{2}\right)$	$X_1, X_2 \in \mathbb{R}$ $a, \hat{a} \in \mathbb{R}$ $s, \hat{s} \in \mathbb{R}_0^+$
11	 $e^{h(\mathbf{a})}$ $\hat{\mathbf{a}}$ $\hat{s}$ $e^{h(s)}$ $\mathcal{N}$ $\mathbf{a}$ $\mathbf{X}_1$ $\mathbf{A}$ $+$ $\mathbf{X}_2$ $f(x_1,x_2,a,s) = \exp\left(-(x_2-ax_1)^2/2s\right)$	Joint coefficient/variance estimation Auto-regression	$e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\left(a \,\middle\vert\, \mathrm{E}[\mathbf{X}_1\mathbf{X}_1^H]^{-1}\mathrm{E}[\mathbf{X}_1X_2], \frac{\mathrm{E}[\mathbf{X}_1\mathbf{X}_1^H]}{\hat{s}}\right)$ $e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle\vert\, -\frac{1}{2}, \frac{\mathrm{E}[(\mathbf{X}_2-\hat{\mathbf{A}}\mathbf{X}_1)^H(\mathbf{X}_2-\hat{\mathbf{A}}\mathbf{X}_1)]}{2}\right)$	$\mathbf{X}_1, \mathbf{X}_2, \mathbf{a} \in \mathbb{R}^n$ $s, \hat{s} \in \mathbb{R}_0^+$ $X_2 = [\mathbf{X}_2]_1$
12	 $\mathbf{A}$ $h(A)$ $\hat{A}$ X_1 $\mathbf{A}$ X_2 $f(x_1,x_2,A) = a_{x_1x_2}$	Finite state machine	$h(\mathbf{A}) = \sum_{x_1,x_2} p(x_1,x_2)\log a_{x_1,x_2}$	$X_1, X_2 \in \mathbf{Z}_n$ $a_{ij} \in [0,1]$

4.9.3 EM and compound nodes

Before applying EM, one may *marginalize* over certain variables. Let us have a look at a small example.

Example 4.3. (EM and compound nodes)
Consider the global function shown in Fig. 4.47(a):

$$f(x_1, x_2, x_3, \theta) \quad \stackrel{\triangle}{=} \quad f_A(\theta) f_B(x_1, x_2, x_3, \theta), \tag{4.235}$$

with

$$f_B(x_1, x_2, x_3, \theta) \stackrel{\triangle}{=} f_{B_1}(x_1) f_{B_2}(x_1, x_2, \theta) f_{B_3}(x_2, x_3, \theta) f_{B_4}(x_3). \tag{4.236}$$

First, we derive "standard" EM (see Fig. 4.47(b)). The two E-log messages h_1 and h_2 are given by:

$$h_1(\theta) \quad = \quad \sum_{x_1, x_2} p(x_1, x_2 | \hat{\theta}^{(k)}) \log f_{B_2}(x_1, x_2, \theta) \tag{4.237}$$

$$h_2(\theta) \quad = \quad \sum_{x_2, x_3} p(x_2, x_3 | \hat{\theta}^{(k)}) \log f_{B_3}(x_2, x_3, \theta), \tag{4.238}$$

where

$$\begin{aligned} &p(x_1, x_2 | \hat{\theta}^{(k)}) \\ &\quad \stackrel{\triangle}{=} \gamma^{-1} \overrightarrow{\mu}(x_1) f_{B_2}(x_1, x_2, \hat{\theta}^{(k)}) \overleftarrow{\mu}(x_2) && (4.239) \\ &\quad = \gamma^{-1} f_{B_1}(x_1) f_{B_2}(x_1, x_2, \hat{\theta}^{(k)}) \sum_{x_3} f_{B_3}(x_2, x_3, \hat{\theta}^{(k)}) f_{B_4}(x_3) && (4.240) \\ &\quad = \gamma^{-1} \sum_{x_3} f_B(x_1, x_2, x_3, \hat{\theta}^{(k)}) && (4.241) \end{aligned}$$

$$\begin{aligned} &p(x_2, x_3 | \hat{\theta}^{(k)}) \\ &\quad \stackrel{\triangle}{=} \gamma^{-1} \overrightarrow{\mu}(x_2) f_{B_3}(x_2, x_3, \hat{\theta}^{(k)}) \overleftarrow{\mu}(x_3) && (4.242) \\ &\quad = \gamma^{-1} \Big(\sum_{x_1} f_{B_1}(x_1) f_{B_2}(x_1, x_2, \hat{\theta}^{(k)}) \Big) f_{B_3}(x_2, x_3, \hat{\theta}^{(k)}) f_{B_4}(x_3) && (4.243) \\ &\quad = \gamma^{-1} \sum_{x_1} f_B(x_1, x_2, x_3, \hat{\theta}^{(k)}), && (4.244) \end{aligned}$$

with

$$\begin{aligned}
\gamma &\triangleq \sum_{x_1,x_2} p(x_1,x_2|\hat{\theta}^{(k)}) && (4.245)\\
&= \sum_{x_2,x_3} p(x_2,x_3|\hat{\theta}^{(k)}) && (4.246)\\
&= \sum_{x_1,x_2,x_3} f_B(x_1,x_2,x_3,\hat{\theta}^{(k)}) && (4.247)\\
&\triangleq f_B(\hat{\theta}^{(k)}), && (4.248)
\end{aligned}$$

and $\overrightarrow{\mu}(x_1)$, $\overrightarrow{\mu}(x_2)$, $\overleftarrow{\mu}(x_2)$ and $\overleftarrow{\mu}(x_3)$ are standard sum-product messages:

$$\begin{aligned}
\overrightarrow{\mu}(x_1) &= f_{B_1}(x_1) && (4.249)\\
\overrightarrow{\mu}(x_2) &= \sum_{x_1} f_{B_1}(x_1) f_{B_2}(x_1,x_2,\hat{\theta}^{(k)}) && (4.250)\\
\overleftarrow{\mu}(x_2) &= \sum_{x_3} f_{B_3}(x_2,x_3,\hat{\theta}^{(k)}) f_{B_4}(x_3) && (4.251)\\
\overleftarrow{\mu}(x_3) &= f_{B_4}(x_3). && (4.252)
\end{aligned}$$

The "standard" EM algorithm alternates the update rules (4.237) and (4.238) with the update rule:

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \left(\log f_A(\theta) + h_1(\theta) + h_2(\theta)\right). \qquad (4.253)$$

Now, we follow a different approach. Before applying EM, we eliminate the variable X_2, resulting in the function $f(x_1, x_3, \theta)$ defined as:

$$f(x_1,x_3,\theta) \triangleq \sum_{x_2} f_{B_2}(x_1,x_2,\theta) f_{B_2}(x_2,x_3,\theta). \qquad (4.254)$$

The h-message out of the (compound) node $f(x_1, x_3, \theta)$ (smallest dashed box in Fig. 4.47(c)) is given by:

$$h(\theta) \triangleq \sum_{x_1,x_3} p(x_1,x_3|\hat{\theta}^{(k)}) \log f(x_1,x_3,\theta). \qquad (4.255)$$

In summary, eliminating the variable X_2 before applying the E-log rule is equivalent to:

a) **Combining** the two nodes f_{B_2} and f_{B_3}.

b) Applying the **E-log rule** to the resulting compound node ("box").

The resulting EM algorithm alternates the update rule (4.255) with the update rule:

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \left(\log f_A(\theta) + h(\theta) \right). \tag{4.256}$$

□

Since the resulting algorithms are (standard) EM algorithms, they are guaranteed to never decrease the global function f (cf. (4.1)).

Deterministic nodes (cf. Remark 4.10) can be handled by combining them with nodes that represent continuous functions ("boxing"), as illustrated in Fig. 4.48 (with $g(x, z, \theta) = \delta(z - f(x, \theta))$). The E-log message h is given by:

$$\begin{aligned} h(\theta) &= \gamma^{-1} \sum_{x,z,z'} \delta(z - f(x, \hat{\theta}))\mu{\uparrow}(z')\mu{\downarrow}(x)h(z, z') \\ &\qquad \cdot \log \sum_{z} \delta(z - f(x, \theta))h(z, z') \qquad (4.257) \\ &= \gamma^{-1} \sum_{x,z'} h(f(x, \hat{\theta}), z')\mu{\downarrow}(x)\mu{\uparrow}(z') \log h(f(x, \theta), z'), \qquad (4.258) \end{aligned}$$

where γ equals:

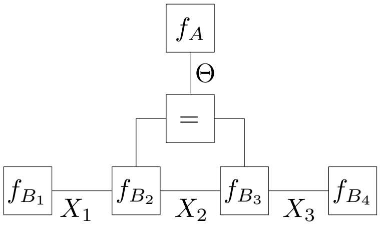
$$\gamma = \sum_{x,z'} h(f(x, \hat{\theta}), z')\mu{\downarrow}(x)\mu{\uparrow}(z'). \tag{4.259}$$

Note that we have applied the generic rule (4.258) in the simple example (4.183), more precisely in (4.192)–(4.193) (see Fig. 4.41).

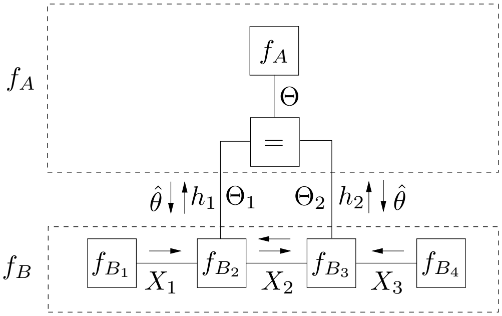
4.9.4 Hybrid EM

In the previous sections, we have described the EM algorithm as an algorithm that performs local computations: it computes messages on a factor graph.

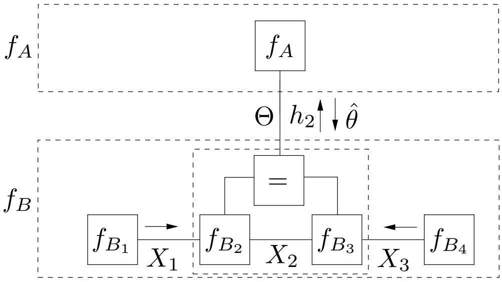
From this "local" perspective, several interesting options become obvious:



(a) Factor graph of (4.254).



(b) Standard EM



(c) Combining the nodes f_{B_2} and f_{B_3}

Figure 4.47: Eliminating variables before applying EM.

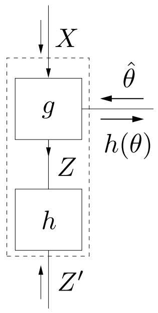


Figure 4.48: Combining the node $g(x, z, \theta) = \delta(z - f(x, \theta))$ with a continuous node h.

- One may apply the E-log update rule in some *subgraph* of the factor graph of f (for estimating some variable Θ); in an other subgraph, a variable Θ' may be estimated by an other message-passing estimation algorithm, such as a sum-product-based gradient algorithm or ICM.

- At a particular *node*, one may combine the E-log update rule with the sum-product rule.

In the following, we investigate the second option, referred to as *hybrid EM* [100]. Consider the generic node depicted in Fig. 4.49. First, we eliminate the variables $Z'_1, \ldots, Z'_m$ by means of the sum-product rule, resulting in the function $f(z_1, \ldots, z_n, \theta_k)$ (dashed box in Fig. 4.49); the variables $Z'_1, \ldots, Z'_m$ are internal variables of the box f. Then, we apply the E-log rule to f.

E-log message out of a box f ("hybrid E-log message") [100]:

$$h(\theta_k) = \sum_{z_1} \cdots \sum_{z_n} p(z_1, \ldots, z_n | \hat{\theta}_k) \log f(z_1, \ldots, z_n, \theta_k), \tag{4.260}$$

$$= \gamma^{-1} \sum_{z_1} \cdots \sum_{z_n} f(z_1, \ldots, z_n, \hat{\theta}_k)\, \mu(z_1) \cdots \mu(z_n) \cdot \log f(z_1, \ldots, z_n, \theta_k), \tag{4.261}$$

with

$$\gamma \triangleq \sum_{z_1} \cdots \sum_{z_n} f(z_1, \ldots, z_n, \hat{\theta}_k)\, \mu(z_1) \cdots \mu(z_n), \tag{4.262}$$

and

$$f(z_1, \ldots, z_n, \theta_k) \propto \sum_{z'_1} \cdots \sum_{z'_m} g(z_1, \ldots, z_n, z'_1, \ldots, z'_m, \theta_k) \cdot \mu(z'_1) \cdots \mu(z'_m), \tag{4.263}$$

where $\mu(z_1), \ldots, \mu(z_n), \mu(z'_1), \ldots, \mu(z'_m)$ are standard sum-product messages.

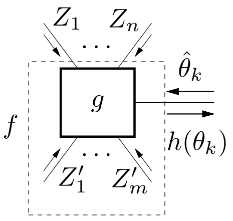


Figure 4.49: Hybrid EM.

The hybrid E-rule (4.261) is of interest for the following reasons:

- A hybrid E-log message may have a **simpler form** than the corresponding E-log message, as we will illustrate by an example at the end of this section (Example 4.6). As a consequence, the maximization (4.177) may be simpler.

- The hybrid E-rule can be applied to **deterministic nodes**, in contrast to the E-log rule (cf. Remark 4.10). An example is depicted in Fig. 4.50 (with $g(x, z, \theta) = \delta(z - f(x, \theta))$). One first eliminates the variable Z by means of the sum-product rule. One applies the E-log rule to the resulting compound node (dashed box in Fig. 4.50). The resulting hybrid E-message h equals:

$$\begin{aligned} h(\theta) &= \gamma^{-1} \sum_{x,z} \delta(z - f(x,\theta)) \mu{\uparrow}(z) \mu{\downarrow}(x) \\ &\qquad \cdot \log \sum_{z} \delta(z - f(x,\theta)) \mu{\uparrow}(z) \qquad (4.264) \\ &= \gamma^{-1} \sum_{x} \mu{\downarrow}(x) \mu{\uparrow}(f(x,\theta)) \log \mu{\uparrow}(f(x,\theta)), \qquad (4.265) \end{aligned}$$

where

$$\gamma = \sum_{x} \mu{\downarrow}(x) \mu{\uparrow}(f(x,\theta)). \qquad (4.266)$$

Often a deterministic node can be handled by combining it with a neighboring continuous node, as described in Section 4.9.3. In some situations, it is nevertheless more attractive to apply the hybrid E-rule instead (cf. Example 4.6).

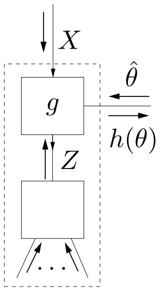


Figure 4.50: Hybrid E-rule applied to a deterministic node $g(x, z, \theta) = \delta(z - f(x, \theta))$.

By a *hybrid EM algorithm*, we denote a modification of the EM algorithm where at some nodes in the subgraph $f_B(x, \theta)$, the hybrid E-rule (4.261) is applied instead of the E-log rule (4.226). The EM algorithm can be viewed as a special case of the hybrid EM algorithm, where the E-log

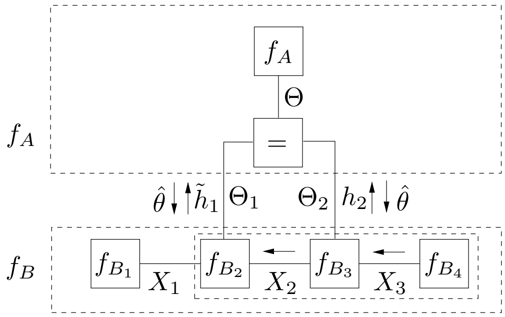


Figure 4.51: A simple hybrid EM algorithm.

rule is applied at *all* nodes in the subgraph $f_B(x, \theta)$. Let us have a look at a simple example of a hybrid EM algorithm.

Example 4.4. (Hybrid EM)
Consider again the global function (4.267):

$$f(x_1, x_2, x_3, \theta) \stackrel{\triangle}{=} f_A(\theta) f_{B_1}(x_1) f_{B_2}(x_1, x_2, \theta) f_{B_3}(x_2, x_3, \theta) f_{B_4}(x_3), \tag{4.267}$$

shown in Fig. 4.47(a).

We apply the hybrid E-rule to the node f_{B_2}: the variable X_2 ("internal variable") is eliminated before the E-log rule is applied to the node f_{B_2}, as illustrated in Fig. 4.51. The hybrid E-message $\tilde{h}_1$ is given by:

$$\begin{aligned} \tilde{h}_1(\theta) &= \sum_{x_1} p(x_1 | \hat{\theta}^{(k)}) \log \sum_{x_2} f_{B_2}(x_1, x_2, \theta) \overleftarrow{\mu}(x_2) & (4.268) \\ &= \sum_{x_1, x_2} p(x_1, x_2 | \hat{\theta}^{(k)}) \log \sum_{x_2} f_{B_2}(x_1, x_2, \theta) \overleftarrow{\mu}(x_2), & (4.269) \end{aligned}$$

where $p(x_1, x_2 | \hat{\theta}^{(k)})$ and $\overleftarrow{\mu}(x_2)$ are given by (4.241) and (4.251) respectively.

At the node f_{B_3}, we apply the E-log rule; the E-log message h_2 is again given by (4.238). The variable X_2 is now treated as a hidden variable. The hybrid EM algorithm alternates the update rules (4.238) and (4.269) with the update rule

$$\hat{\theta}^{(k+1)} = \underset{\theta}{\operatorname{argmax}} \Big(\log f_A(\theta) + \tilde{h}_1(\theta) + h_2(\theta) \Big). \tag{4.270}$$

Note that the variable X_2 is thus treated *simultaneously* as an internal variable (at node f_{B_2}) and as a hidden variable (at node f_{B_3}), in contrast to the approach of Fig. 4.47(c), where the variable X_2 is *consistently* treated as an internal variable (of the box $f(x_1, x_3, \theta)$). Whereas the approach depicted in Fig. 4.47(c) is guaranteed to decrease the global function f at each iteration, there is no such guarantee for the hybrid EM algorithm. However, we are able to characterize the fixed points of the algorithm. □

Theorem 4.4. Assume that a factor graph of a global function $f(x,\theta) \triangleq f_A(\theta)f_B(x,\theta)$ is available whose subgraph $f_B(x,\theta)$ is *cycle-free.* The fixed points of a hybrid EM algorithm applied on that factor graph are stationary points of the marginal $f(\theta)$.

Proof: It is instructive to first prove the theorem for the simple hybrid EM algorithm (4.270). We consider the general case afterwards.

Note first of all that the hybrid EM algorithm (4.270) operates on a factor graph whose subgraph $f_B(x,\theta)$ is cycle-free (see Fig. 4.47(a)). By defining the function $\tilde{f}(\theta,\theta')$ as

$$\tilde{f}(\theta,\theta') \triangleq \log f_A(\theta) + \tilde{f}_1(\theta,\theta') + \tilde{f}_2(\theta,\theta'), \tag{4.271}$$

with

$$\tilde{f}_1(\theta,\theta') \triangleq \sum_{x_1,x_2} p(x_1,x_2|\theta') \log \sum_{x_2} f_{B_2}(x_1,x_2,\theta) \overleftarrow{\mu}(x_2) \tag{4.272}$$

$$\tilde{f}_2(\theta,\theta') \triangleq \sum_{x_2,x_3} p(x_2,x_3|\theta') \log f_{B_3}(x_2,x_3,\theta), \tag{4.273}$$

the rule (4.270) can be rewritten as

$$\hat{\theta}^{(k+1)} = \operatorname*{argmax}_{\theta} \tilde{f}(\theta,\theta^{(k)}). \tag{4.274}$$

The fixed points of the hybrid EM algorithm are implicitly defined by the equation:

$$\hat{\theta}^{\text{fixed}} = \operatorname*{argmax}_{\theta} \tilde{f}(\theta,\theta^{\text{fixed}}). \tag{4.275}$$

As a consequence of (4.275), the fixed points fulfill the constraint:

$$\nabla_\theta \tilde{f}(\theta,\theta^{\text{fixed}})\Big|_{\theta^{\text{fixed}}} = 0. \tag{4.276}$$

We now show that

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} \;=\; \nabla_\theta \log f(\theta)|_{\theta'}\,. \tag{4.277}$$

First, note that

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} = \nabla_\theta \log f_A(\theta)|_{\theta'} + \nabla_\theta \tilde{f}_1(\theta,\theta')\Big|_{\theta'} + \nabla_\theta \tilde{f}_2(\theta,\theta')\Big|_{\theta'}. \tag{4.278}$$

We rewrite the second term in the RHS of (4.278) as:

$$\begin{aligned}
&\nabla_\theta\ \tilde{f}_1(\theta,\theta')\Big|_{\theta'}\\
&= \sum_{x_1,x_2} p(x_1,x_2|\theta')\ \nabla_\theta \log \sum_{x_2} f_{B_2}(x_1,x_2,\theta)\,\overleftarrow{\mu}(x_2)\Bigg|_{\theta'} &(4.279)\\
&= \sum_{x_1,x_2} p(x_1,x_2|\theta') \frac{\sum_{x_2} \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_2)}{\sum_{x_2} f_{B_2}(x_1,x_2,\theta')\,\overleftarrow{\mu}(x_2)}. &(4.280)
\end{aligned}$$

We substitute (4.241) and (4.248) in (4.280) and obtain:

$$\begin{aligned}
&\nabla_\theta\ \tilde{f}_1(\theta,\theta')\Big|_{\theta'}\\
&= \frac{1}{f_B(\theta')} \sum_{x_1,x_2} f_{B_2}(x_1,x_2,\theta')\,\overrightarrow{\mu}(x_1)\,\overleftarrow{\mu}(x_2)\\
&\quad \cdot \frac{\sum_{x_2} \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_2)}{\sum_{x_2} f_{B_2}(x_1,x_2,\theta')\,\overleftarrow{\mu}(x_2)} &(4.281)\\
&= \frac{1}{f_B(\theta')} \sum_{x_1} \overrightarrow{\mu}(x_1)\Big(\sum_{x_2} f_{B_2}(x_1,x_2,\theta')\,\overleftarrow{\mu}(x_2)\Big)\\
&\quad \cdot \frac{\sum_{x_2} \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_2)}{\sum_{x_2} f_{B_2}(x_1,x_2,\theta')\,\overleftarrow{\mu}(x_2)} &(4.282)\\
&= \frac{1}{f_B(\theta')} \sum_{x_1} \overrightarrow{\mu}(x_1) \sum_{x_2} \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_2) &(4.283)\\
&= \frac{1}{f_B(\theta')} \sum_{x_1,x_2} \overrightarrow{\mu}(x_1)\ \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_2). &(4.284)
\end{aligned}$$

By substituting (4.249) and (4.251) in (4.284), it follows:

$$\nabla_\theta\ \tilde{f}_1(\theta,\theta')\Big|_{\theta'}$$

$$= \frac{1}{f_B(\theta')} \sum_{x_1,x_2} f_{B_1}(x_1)\ \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'} \cdot \Big(\sum_{x_3} f_{B_3}(x_2,x_3,\theta') f_{B_4}(x_3) \Big) \tag{4.285}$$

$$= \frac{1}{f_B(\theta')} \sum_{x_1,x_2,x_3} f_{B_1}(x_1)\ \nabla_\theta f_{B_2}(x_1,x_2,\theta)|_{\theta'}\ f_{B_3}(x_2,x_3,\theta') f_{B_4}(x_3) \tag{4.286}$$

$$= \frac{1}{f_B(\theta')} \nabla_\theta \Big(\sum_{x_1,x_2,x_3} f_{B_1}(x_1) f_{B_2}(x_1,x_2,\theta) f_{B_3}(x_2,x_3,\theta') f_{B_4}(x_3) \Big) \Big|_{\theta'} \tag{4.287}$$

$$= \sum_{x_1,x_2} \nabla_\theta\ p(x_1,x_2|\theta)|_{\theta'} \tag{4.288}$$

$$= \mathrm{E}_{p(x_1,x_2|\theta)} \left[\nabla_\theta\ \log p(x_1,x_2|\theta)|_{\theta'} \right]. \tag{4.289}$$

Note that (4.251) and hence (4.285)–(4.289) hold since the graph of $f_B(x,\theta)$ is *cycle-free.*

Similarly, the third term in the RHS of (4.271) can be written as

$$\nabla_\theta\ \tilde{f}_2(\theta,\theta')\Big|_{\theta'}$$

$$= \frac{1}{f_B(\theta')} \sum_{x_2,x_3} \overrightarrow{\mu}(x_2)\ \nabla_\theta f_{B_3}(x_2,x_3,\theta)|_{\theta'}\ \overleftarrow{\mu}(x_3) \tag{4.290}$$

$$= \frac{1}{f_B(\theta')} \sum_{x_1,x_2,x_3} f_{B_1}(x_1) f_{B_2}(x_1,x_2,\theta')\ \nabla_\theta f_{B_3}(x_2,x_3,\theta)|_{\theta'}\ f_{B_4}(x_3) \tag{4.291}$$

$$= \sum_{x_2,x_3} \nabla_\theta\ p(x_2,x_3|\theta)|_{\theta'} \tag{4.292}$$

$$= \mathrm{E}_{p(x_2,x_3|\theta)} \left[\nabla_\theta\ \log p(x_2,x_3|\theta)|_{\theta'} \right]. \tag{4.293}$$

Note that:

$$\begin{aligned} \nabla_\theta f_B(x,\theta) =& f_{B_1}(x_1) \nabla_\theta f_{B_2}(x_1,x_2,\theta) f_{B_3}(x_2,x_3,\theta) f_{B_4}(x_3) \\ &+ f_{B_1}(x_1) f_{B_2}(x_1,x_2,\theta) \nabla_\theta f_{B_3}(x_2,x_3,\theta) f_{B_4}(x_3). \end{aligned} \tag{4.294}$$

Therefore, by substituting (4.286) and (4.291) in (4.271), we obtain:

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} = \nabla_\theta \log f_A(\theta)|_{\theta'} + \frac{1}{f_B(\theta')} \sum_x \nabla_\theta f_B(x,\theta)|_{\theta'} \tag{4.295}$$

$$= \nabla_\theta \log f_A(\theta)|_{\theta'} + \frac{1}{f_B(\theta')} \nabla_\theta f_B(\theta)|_{\theta'} \tag{4.296}$$
$$= \nabla_\theta \log f_A(\theta)|_{\theta'} + \nabla_\theta \log f_B(\theta)|_{\theta'} \tag{4.297}$$
$$= \nabla_\theta \log f(\theta)|_{\theta'}, \tag{4.298}$$

where $f_B(\theta) \triangleq \sum_x f_B(x,\theta)$. From (4.276) and (4.298) it follows:

$$\nabla_\theta f(\theta)|_{\theta^{\text{fixed}}} = 0, \tag{4.299}$$

or, in words: the fixed points θ^{fixed} of the hybrid EM algorithm (4.270) are stationary points of the marginal $f(\theta) \triangleq \sum_x f(x,\theta)$ of the global function $f(x,\theta)$ (4.267).

The key point in the above proof is the fact that the terms (4.286) and (4.291) have the same form, since it leads to the equalities (4.298) and (4.299). Note that the term (4.286) originates from a hybrid E-message, whereas the term (4.291) originates from an E-log message. If we had applied the E-log rule (instead of the hybrid E-rule) to the node f_{B_2} (cf. (4.237)), we would also then have obtained the term (4.286).

We now consider the generic node f_{B_k} depicted in Fig. 4.52, where the variables Z_ℓ and Z'_ℓ are certain components of the vector X. The corresponding term $\tilde{f}_k(\theta,\theta')$ (cf. (4.272) and (4.273)) equals:

$$\tilde{f}_k(\theta,\theta') = \sum_{z,z'} p(z,z'|\theta') \log \sum_{z'} f_{B_k}(z,z',\theta) \prod_{\ell=1}^{m} \mu(z'_\ell), \tag{4.300}$$

where $Z \triangleq (Z_1,\ldots,Z_n)$, and $Z' \triangleq (Z'_1,\ldots,Z'_m)$. Therefore,

$$\nabla_\theta \tilde{f}_k(\theta,\theta')\Big|_{\theta'}$$
$$= \sum_{z,z'} p(z,z'|\theta')\, \nabla_\theta \log \sum_{z'} f_{B_k}(z,z',\theta) \prod_{\ell=1}^{m} \mu(z'_\ell)\Bigg|_{\theta'} \tag{4.301}$$
$$= \sum_{z,z'} p(z,z'|\theta') \frac{\sum_{z'} \nabla_\theta f_{B_k}(z,z',\theta)|_{\theta'} \prod_{\ell=1}^{m} \mu(z'_\ell)}{\sum_{z'} f_{B_k}(z,z',\theta') \prod_{\ell=1}^{m} \mu(z'_\ell)} \tag{4.302}$$
$$= \frac{1}{f_B(\theta')} \sum_{z,z'} \nabla_\theta f_{B_k}(z,z',\theta)|_{\theta'} \prod_{\ell=1}^{n} \mu(z_\ell) \prod_{\ell=1}^{m} \mu(z'_\ell). \tag{4.303}$$

If the subgraph $f_B(x,\theta)$ is cycle-free, we can rewrite (4.303) as:

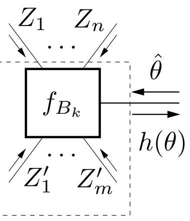


Figure 4.52: Hybrid EM.

$$\nabla_\theta \tilde{f}_k(\theta,\theta')\Big|_{\theta'} = \frac{1}{f_B(\theta')}\sum_x \frac{\nabla_\theta f_{B_k}(z,z',\theta)|_{\theta'}}{f_{B_k}(z,z',\theta')} f_B(x,\theta'). \tag{4.304}$$

Note that the terms (4.286) and (4.291) have the form (4.304). In the term (4.286), $Z = X_1$ and $Z' = X_2$, whereas in the term (4.291), the hidden variables are given by $Z = (X_2, X_3)$ and there are no internal variables Z'. The expression (4.304) holds for any choice of hidden variables Z and internal variables Z'. Note also that the expression (4.304) makes no distinction between hidden variables and internal variables. In particular, if both Z and Z' are treated as hidden variables (as in standard EM), the expression (4.304) remains unchanged. Each node f_{B_k} amounts thus to a term of the form (4.304); therefore, the equalities (4.295)–(4.299) hold for *general* cycle-free graphs $f_B(x,\theta)$. In conclusion: the fixed points θ^{fixed} of a general hybrid EM algorithm (operating on a factor graph of $f(x,\theta) \triangleq f_A(\theta)f_B(x,\theta)$ whose subgraph $f_B(x,\theta)$ is cycle-free) are stationary points of the marginal $f(\theta) \triangleq \sum_x f(x,\theta)$.
□

Theorem 4.4 concerns *cycle-free* subgraphs $f_B(x,\theta)$. What if the subgraph $f_B(x,\theta)$ is *cyclic*? For cyclic graphs, Theorem 4.4 does not hold, i.e., the fixed points of hybrid EM algorithms applied to cyclic subgraphs $f_B(x,\theta)$ are *not* equal to the stationary points of the marginal $f(\theta)$. But, if the fixed points are not the stationary points of the *exact* marginal $f(\theta)$, are they perhaps stationary points of the *approximate* marginal $b(\theta)$ obtained by iterative sum-product message passing on the subgraph $f_B(x,\theta)$? We now show by a simple example that the answer to that question is "no".

Example 4.5. ((Hybrid) EM on cyclic subgraphs $f_B(x,\theta)$)

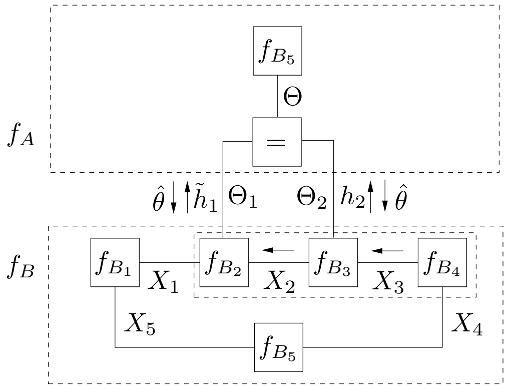


Figure 4.53: A simple hybrid EM algorithm operating on a cyclic subgraph $f_B(x, \theta)$.

We consider the global function shown in Fig. 4.53:

$$f(x_1, x_2, x_3, x_4, x_5, \theta) \quad \stackrel{\triangle}{=} \quad f_A(\theta) f_B(x_1, x_2, x_3, x_4, x_5, \theta), \quad (4.305)$$

with

$$f_B(x_1, x_2, x_3, x_4, x_5, \theta) \quad \stackrel{\triangle}{=} \quad f_{B_1}(x_1, x_5) f_{B_2}(x_1, x_2, \theta) f_{B_3}(x_2, x_3, \theta) \cdot f_{B_4}(x_3, x_4) f_{B_5}(x_4, x_5). \quad (4.306)$$

Note that the subgraph $f_B(x, \theta)$ is cyclic, in contrast to the subgraph $f_B(x, \theta)$ in Fig. 4.51. As in Example 4.4, we apply the hybrid E-rule to the node f_{B_2} and the E-log rule to the node f_{B_3}. We are interested in the fixed points of the resulting algorithm. The algorithm may, however, not converge and thus never attain a fixed point.

We follow the line of thought in Example 4.4; we again again start from the function $\tilde{f}(\theta, \theta')$ (4.271), which contains the terms $\tilde{f}_1(\theta, \theta')$ (4.272) and $\tilde{f}_2(\theta, \theta')$ (4.273). Note that the equalities (4.284) and (4.290) also hold for the factor graph of Fig. 4.53. The sum-product messages $\overrightarrow{\mu}(x_1)$, $\overrightarrow{\mu}(x_2)$, $\overleftarrow{\mu}(x_2)$, and $\overleftarrow{\mu}(x_3)$ in (4.284) and (4.290) are now computed *iteratively*, since the subgraph f_B is cyclic; they are in general *not* given by (4.249)–(4.252) respectively. As a consequence, the equalities (4.286) and (4.291) do *not* hold. It follows:

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} = \nabla_\theta \log f_A(\theta)|_{\theta'} + \gamma^{-1}\sum_{x_1,x_2} \overrightarrow{\mu}(x_1)\overleftarrow{\mu}(x_2)\nabla_\theta\, f_{B_2}(x_1,x_2,\theta)|_{\theta'} + \gamma^{-1}\sum_{x_2,x_3} \overrightarrow{\mu}(x_2)\overleftarrow{\mu}(x_3)\nabla_\theta\, f_{B_3}(x_2,x_3,\theta)|_{\theta'}\,, \tag{4.307}$$

where

$$\gamma \triangleq \sum_{x_1,x_2} \overrightarrow{\mu}(x_1)\overleftarrow{\mu}(x_2) f_{B_2}(x_1,x_2,\theta') \tag{4.308}$$

$$= \sum_{x_2,x_3} \overrightarrow{\mu}(x_2)\overleftarrow{\mu}(x_3) f_{B_3}(x_2,x_3,\theta'). \tag{4.309}$$

In analogy to (4.289) and (4.293), we can rewrite (4.307) as:

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} = \nabla_\theta\, \log f_A(\theta)|_{\theta'} + \mathrm{E}_{b(x_1,x_2|\theta')}\left[\nabla_\theta \log f(x_1,x_2,\theta)|_{\theta'}\right] + \mathrm{E}_{b(x_2,x_3|\theta')}\left[\nabla_\theta \log f(x_2,x_3,\theta)|_{\theta'}\right], \tag{4.310}$$

where

$$b(x_1,x_2|\theta') \triangleq \gamma^{-1} \sum_{x_1,x_2} \overrightarrow{\mu}(x_1)\overleftarrow{\mu}(x_2) f_{B_2}(x_1,x_2,\theta') \tag{4.311}$$

$$b(x_2,x_3|\theta') \triangleq \gamma^{-1} \sum_{x_2,x_3} \overrightarrow{\mu}(x_2)\overleftarrow{\mu}(x_3) f_{B_3}(x_2,x_3,\theta'). \tag{4.312}$$

Note that we write $b(\cdot|\cdot)$ and not $p(\cdot|\cdot)$ in (4.311)–(4.312), since the expressions (4.311)–(4.312) are not the *exact* marginals but approximations ("beliefs"). If we define the function $\log \hat{f}(\theta)$ as:

$$\log \hat{f}(\theta) \triangleq \log f_A(\theta) + \int_{-\infty}^{\theta} \mathrm{E}_{b(x_1,x_2|\tilde{\theta})}\left[\nabla_\theta \log f(x_1,x_2,\tilde{\theta})\right] d\tilde{\theta} + \int_{-\infty}^{\theta} \mathrm{E}_{b(x_2,x_3|\tilde{\theta})}\left[\nabla_\theta \log f(x_2,x_3,\tilde{\theta})\right] d\tilde{\theta}, \tag{4.313}$$

then

$$\nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'} = \nabla_\theta \log \hat{f}(\theta)\Big|_{\theta'}. \tag{4.314}$$

From (4.276), it follows that the fixed points of the hybrid EM algorithm are stationary points of the function $\hat{f}(\theta)$, where the beliefs occurring in $\hat{f}(\theta)$ (cf. (4.311)–(4.313)) are computed by means of the sum-product messages $\overrightarrow{\mu}(x_1)$, $\overrightarrow{\mu}(x_2)$, $\overleftarrow{\mu}(x_2)$, and $\overleftarrow{\mu}(x_3)$ available at *convergence* of the sum-product algorithm. Note that $\hat{f}(\theta)$ is *not* equal to the marginal $b(\theta)$ obtained by iterative sum-product message passing. The latter is given by:

$$b(\theta) \propto f_A(\theta)\mu_{f_{B_2}\to\Theta}(\theta)\mu_{f_{B_3}\to\Theta}(\theta), \tag{4.315}$$

where

$$\mu_{f_{B_2}\to\Theta}(\theta) \propto \sum_{x_1,x_2} f_{B_2}(x_1,x_2,\theta)\,\overrightarrow{\mu}(x_1)\,\overleftarrow{\mu}(x_2) \tag{4.316}$$

$$\mu_{f_{B_3}\to\Theta}(\theta) \propto \sum_{x_2,x_3} f_{B_3}(x_2,x_3,\theta)\,\overrightarrow{\mu}(x_2)\,\overleftarrow{\mu}(x_3). \tag{4.317}$$

Since the sum-product messages $\overrightarrow{\mu}(x_1)$, $\overrightarrow{\mu}(x_2)$, $\overleftarrow{\mu}(x_2)$ and $\overleftarrow{\mu}(x_3)$ depend on θ, we have in general:

$$\nabla_\theta \log b(\theta)\big|_{\theta'} \neq \nabla_\theta \tilde{f}(\theta,\theta')\Big|_{\theta'}, \tag{4.318}$$

where the RHS is given by (4.310). Therefore, the fixed points of the hybrid EM-algorithm are not stationary points of $b(\theta)$.

If the subgraph $f_B(x,\theta)$ is *cycle-free*, the beliefs $b(\cdot|\theta)$ are equal to the exact marginals $p(\cdot|\theta)$, hence, $\hat{f}(\theta) = f(\theta)$. The fixed points of the hybrid EM-algorithm are then stationary points of $f(\theta)$.

Since the E-log rule is a special case of the hybrid E-rule, the above exposition also applies to the standard EM-algorithm (with h-messages (4.237) and (4.238)) applied on the graph of Fig. 4.53. $\square$

From the previous example, it is but a small step to the following theorem.

Theorem 4.5. Assume that a factor graph of a global function $f(x,\theta) \triangleq f_A(\theta)f_B(x,\theta)$ is available (whose subgraph $f_B(x,\theta)$ may be cycle-free or cyclic). The fixed points of a (hybrid) EM algorithm applied on that factor graph are stationary points of the function $\hat{f}(\theta)$, defined as:

$$\log \hat{f}(\theta) \triangleq \log f_A(\theta) + \int_{-\infty}^{\theta} \mathrm{E}_{b(x|\tilde{\theta})}\left[\nabla_\theta \log f_B(x,\tilde{\theta})\right] d\tilde{\theta}, \tag{4.319}$$

where the beliefs $b(\cdot|\theta)$ are computed by means of the sum-product messages available at convergence of the sum-product algorithm.

Example 4.6. (AR coefficient estimation)
We consider the system depicted in Fig. 4.54(a); we wish to estimate the

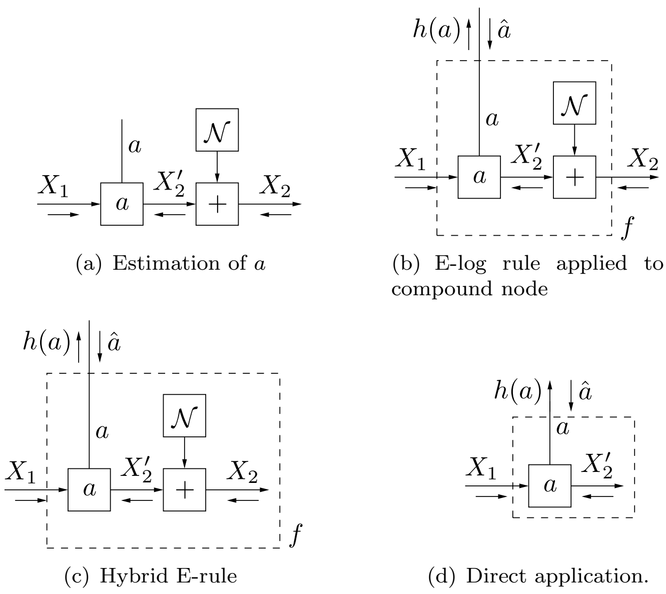


Figure 4.54: State transition node for scalar AR coefficient estimation.

coefficient a by an EM-type algorithm. The messages along the edges X_1, X_2, and X_2' are assumed to be Gaussian.

In the following, we outline two approaches. In the first approach, the E-log rule (4.226) is applied to the compound node f of Fig. 4.54(b) defined as:

$$f(x_1, x_2, a) \triangleq \frac{1}{\sqrt{2\pi s}} e^{(x_2 - a x_1)^2/s}. \tag{4.320}$$

The message $e^{h(a)}$ leaving the box $f(x_1, x_2, a)$ is given by Rule 7 of Table 4.2:

$$e^{h(a)} \propto \mathcal{N}^{-1}\left(a \,\middle|\, \frac{\mathrm{E}[X_1 X_2]}{\mathrm{E}[X_1^2]}, \frac{\mathrm{E}[X_1^2]}{s}\right). \tag{4.321}$$

In the second approach, we apply the hybrid E-rule (4.261), as illustrated in Fig. 4.54(c). First, we integrate over the variable X_2', amounting to:

$$f(x_1, a) \stackrel{\triangle}{=} \int g(x_1, x_2, a)\mu_{X_2' \to g}(x_2)dx_2' \tag{4.322}$$
$$= \int \delta(x_2 - ax_1)\mu_{X_2' \to g}(x_2)dx_2' \tag{4.323}$$
$$= \mu_{X_2' \to g}(ax_1) \tag{4.324}$$
$$\stackrel{\triangle}{=} \mathcal{N}(ax_1 \mid m_2', v_2')\,. \tag{4.325}$$

The E-log message leaving the box $f(x_1, a)$ is given by:

$$h'(a) = \int_{x_1} p(x_1|\hat{a}^{(k)}) \log f(x_1, a)\, dx_1 \tag{4.326}$$
$$= \int_{x_1} p(x_1|\hat{a}^{(k)}) \log \mu_{X_2' \to f}(ax_1)\, dx_1 \tag{4.327}$$
$$= C - \frac{1}{2v_2'}\left(a^2\mathrm{E}[X_1^2|\hat{a}^{(k)}] - 2am_2'\mathrm{E}[X_1|\hat{a}^{(k)}] + (m_2')^2\right) \tag{4.328}$$

with

$$C = -\frac{1}{2}\log(2\pi v_2') \tag{4.329}$$

and in the exponential domain

$$e^{h'(a)} \propto \mathcal{N}^{-1}\left(a \;\middle|\; \frac{m_2'\mathrm{E}[X_1]}{\mathrm{E}[X_1^2]}, \frac{\mathrm{E}[X_1^2]}{v_2'}\right) \tag{4.330}$$

where the expectations are w.r.t. the distribution $p(x_1|\hat{a}^{(k)})$. Note that the message (4.330) follows directly from the message (4.321), where $X_2 \stackrel{\triangle}{=} m_2'$ and where $s \stackrel{\triangle}{=} v_2'$. Note also that the message (4.330) is easier to evaluate than the message (4.321).

In certain applications, the multiplication node occurs in absence of the addition node, as illustrated in Fig. 4.54(d). This situation occurs for example in time-dependent linear systems *without* input noise (cf. Fig. H.1 with $\mathbf{B} = 0$ and time-dependent $\mathbf{A}$-matrices). The hybrid E-rule (4.330) can be applied directly to the multiplication node, in contrast to the rule (4.321).

The extension to the vector case (autoregression), as shown in Fig. 4.55,

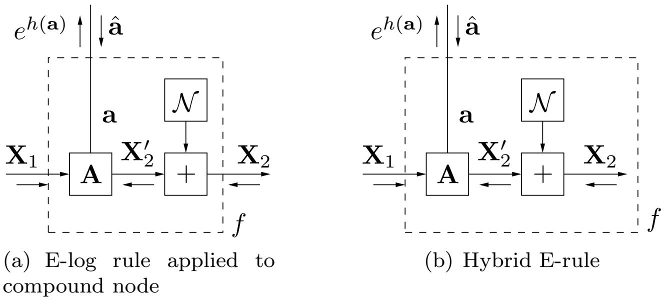


Figure 4.55: Autoregression.

is straightforward.[11] The dashed box in Fig. 4.55(a) represents the factor:

$$f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-1/2(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^H \mathbf{V}^{-1} (\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)\right), \tag{4.331}$$

with

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{a}^H \\ \mathbf{I} \quad \mathbf{0} \end{bmatrix}, \tag{4.332}$$

and $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^n$.

The message $e^{h(\mathbf{a})}$ leaving the box $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a})$ equals (cf. Section J.3.2):

$$e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\left(\mathbf{a} \,\middle|\, \mathbf{m}_a, \mathbf{W}_a\right), \tag{4.333}$$

where

$$\mathbf{W}_a = w_{11} \mathrm{E}\left[\mathbf{X}_1 \mathbf{X}_1^H\right] \tag{4.334}$$

$$\mathbf{m}_a = \mathbf{W}_a^{-1} \left(\sum_{k=1}^{n} w_{1k} \mathrm{E}\left[\mathbf{X}_1 \left[\mathbf{X}_2\right]_k\right] - \sum_{k=1}^{n-1} w_{1k+1} \mathrm{E}\left[\mathbf{X}_1 \left[\mathbf{X}_1\right]_k\right] \right), \tag{4.335}$$

with $[\mathbf{X}_i]_j$ the j-th component of the (random) vector $\mathbf{X}_i$, and w_{ij} the element (i, j) of $\mathbf{W} \triangleq \mathbf{V}^{-1}$ $(i, j = 1, \ldots, n)$. Note that the expression $\mathrm{E}\left[\mathbf{X}_1 \left[\mathbf{X}_1\right]_k\right]$ stands for a vector whose ℓ-th component is given by $\mathrm{E}\left[\left[\mathbf{X}_1\right]_\ell \left[\mathbf{X}_1\right]_k\right]$.

[11] We will denote vectors in bold in order to avoid confusion between the *scalar* case and the *vector* case.

The E-log message leaving the box $f(\mathbf{x}_1, \mathbf{a})$ in Fig. 4.55(b) is given by:

$$e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\Big(\mathbf{a} \,\Big|\, \mathbf{m}_a, \mathbf{W}_a\Big) \tag{4.336}$$

with

$$\mathbf{W}_a = w_{11}\mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right] \tag{4.337}$$

$$\mathbf{m}_a = \mathbf{W}_a^{-1}\Big(\sum_{k=1}^{n} w_{1k}\,[\mathbf{m}_2']_k\,\mathrm{E}\left[\mathbf{X}_1\right] - \sum_{k=1}^{n-1} w_{1k+1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_1\right]_k\right]\Big), \tag{4.338}$$

where $\mathbf{m}_2' \in \mathbb{R}^n$ and $\mathbf{V}_2' \in \mathbb{R}^{n\times n}$ is the mean and covariance matrix of the incoming message along the edge X_2', and w_{ij} is the element (i,j) of the matrix $\mathbf{W}_2 \triangleq \mathbf{V}_2'^{-1}$ $(i,j = 1,\ldots,n)$. The message (4.336) follows directly from the message (4.333), where the variable $\mathbf{X}_2$ is set to $\mathbf{m}_2$ and where $\mathbf{V} \triangleq \mathbf{V}_2'$. □

4.9.5 Extensions of EM

The evaluation of the upward message $h(\theta)$ (4.174) (E-step) and/or the downward message $\hat{\theta}^{(k)}$ (4.177) (M-step) is sometimes not feasible. In the literature, several methods have been proposed to solve this problem.

If the computation of the message $h(\theta)$ is not tractable, one may compute $h(\theta)$ approximately by means of Monte-Carlo methods, an approach called **Monte-Carlo EM** (MCEM) [128, p. 214–216]. The sum-product messages $\mu(z)$ (cf. (4.226)) are then represented as lists of samples. The special case where the messages $\mu(z)$ are represented by a *single* sample is called **Stochastic EM** (SEM) [128, p. 216–218].

If the computation of $\hat{\theta}^{(k)}$ (4.177) is problematic, one may carry out the maximization in (4.177) by ICM (cf. Section 4.8.2), an approach called **Expectation Conditional Maximization** (ECM) [128, p. 167–171]. As usual, several variables may be grouped in the ICM steps. In addition, several update schedules are possible; one may update the h-messages after each (or a certain number of) ICM step(s) ("multicycle ECM" [128, p. 214–216]). Alternatively, if the computation of the messages $h(\theta)$ is expensive, one may only update the h-messages after the ICM algorithm has converged to an estimate $\hat{\theta}^{(k)}$. Some of the ICM steps may be replaced by a *conditional* maximization of the marginal $f(\theta)$, an approach called **Expectation Conditional Maximization Either** (ECME) [128, p. 206–208]; one maximizes the marginal $f(\theta)$ w.r.t. certain components θ_k of the parameter vector θ while the other components are kept *fixed* at the current estimate. In an extension of ECM called **Alternating ECM** (AECM) [128, p. 183–184], the hidden variable(s) may be chosen differently at each ICM iteration. ECME may be regarded as a special case of AECM, where in particular ICM steps, there are *no* hidden variables. Another special case of ACME is the **Space Alternating Generalized EM** algorithm (SAGE) [128, p. 206]; in SAGE, no variables are grouped in the ICM steps.

Instead of ICM, gradient methods may be used to (approximately) carry out the M-step. This approach is called **gradient EM** [128, p. 156–158]); it is the subject of the next section.

4.9.6 Gradient EM

The maximization in the RHS of (4.177) is sometimes intractable. The estimate $\hat{\theta}^{(k+1)}$ may be determined by gradient methods.

Let us have a look at a simple example. Suppose that

$$f_A(\theta) \stackrel{\triangle}{=} f_{A_1}(\theta_1)f_{A_2}(\theta_1,\theta_2)\ldots f_{A_n}(\theta_{n-1},\theta_n), \tag{4.339}$$

and

$$\begin{aligned} f_B(x,\theta) \;&\stackrel{\triangle}{=}\; f_{B_0}(x_0)f_{B_1}(x_0,x_1,y_1,\theta_1)f_{B_2}(x_1,x_2,y_2,\theta_2)\\ &\quad \ldots f_{B_n}(x_{n-1},x_n,y_n,\theta_n), \end{aligned} \tag{4.340}$$

as illustrated in Fig. 4.40. As we pointed out before, the probability function $f(x,\theta)$ may represent a state-space model parameterized by the parameter θ, whose prior model is determined by $f_A(\theta)$. In this case, the downward message (4.177) equals

$$\begin{aligned} (\hat{\theta}_1,\ldots,\hat{\theta}_n)^T \;&=\; \operatorname*{argmax}_{\theta_1,\ldots,\theta_n}\big(\log f_{A_1}(\theta_1)+\log f_{A_2}(\theta_1,\theta_2)+\ldots\\ &\qquad +\log f_{A_n}(\theta_{n-1},\theta_n)+h_1(\theta_1)+h_n(\theta_n)\big) \qquad (4.341)\\ &=\; \operatorname*{argmax}_{\theta_1,\ldots,\theta_n}\big(f_{A_1}(\theta_1)f_{A_2}(\theta_1,\theta_2)\ldots f_{A_n}(\theta_{n-1},\theta_n)\ldots\\ &\qquad \cdot\, e^{h_1(\theta_1)}\ldots e^{h_n(\theta_n)}\big), \qquad (4.342) \end{aligned}$$

where

$$h_k(\theta_k) = \sum_{x_{k-1}} \sum_{x_k} p_{\mathrm{B}}(x_{k-1}, x_k | y, \hat{\theta}) \log f_k(x_{k-1}, x_k, y_k, \theta_k), \tag{4.343}$$

and

$$p_{\mathrm{B}}(x_{k-1}, x_k | y, \hat{\theta}) = \frac{f_k(x_{k-1}, x_k, y_k, \hat{\theta})\, \mu_{X_{k-1}\to f_k}(x_{k-1})\, \mu_{X_k \to f_k}(x_k)}{\sum_{x_{k-1}} \sum_{x_k} f_{B_k}(x_{k-1}, x_k, y, \hat{\theta})\, \mu_{X_{k-1}\to f_k}(x_{k-1})\, \mu_{X_k\to f_k}(x_k)}. \tag{4.344}$$

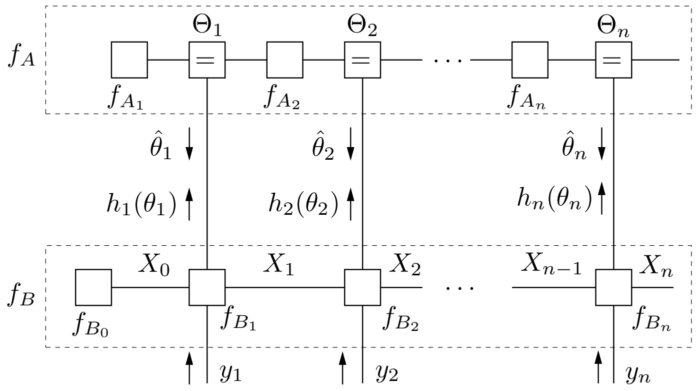


Figure 4.56: Factor graph of (4.339) and (4.340).

The gradient $\nabla_\theta h(\theta) \stackrel{\triangle}{=} \big(\nabla_{\theta_1} h(\theta), \ldots, \nabla_{\theta_n} h(\theta)\big)^T$ required for steepest descent is computed as follows

$$\nabla_{\theta_\ell} h_\ell(\theta_\ell)$$

$$= \nabla_{\theta_\ell} \Big[\sum_{x_{\ell-1}, x_\ell} p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \log f_{B_\ell}(x_{\ell-1}, x_\ell, y, \theta_\ell) \Big], \tag{4.345}$$

$$= \sum_{x_{\ell-1}, x_\ell} p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta}) \nabla_{\theta_\ell} \log f_{B_\ell}(x_{\ell-1}, x_\ell, y, \theta_\ell). \tag{4.346}$$

Note that (4.344) and hence also the rule (4.346) involve standard sum-product messages. Those messages may again be represented in different ways, such as lists of particles, quantized messages, Gaussian distributions etc.

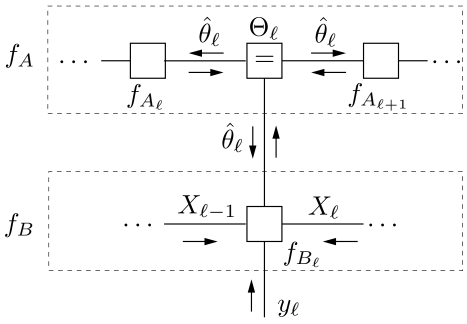


Figure 4.57: Steepest descent as summary propagation.

Expectation maximization, in which the M-step is performed by steepest descent, may then be formulated as follows (see Fig. 4.57):

a) The equality constraint nodes Θ_ℓ broadcast the estimates $\hat{\theta}_\ell^{(k)}$.

b) The nodes f_{A_ℓ} and $f_{A_{\ell+1}}$ reply with the messages $\nabla_{\theta_\ell} \log f_{A_\ell}|_{\hat{\theta}^{(k)}}$ and $\nabla_{\theta_\ell} \log f_{A_{\ell+1}}\big|_{\hat{\theta}^{(k)}}$ respectively.

c) A forward and backward sum(mary)-product sweep is performed in the box f_B.

d) The nodes f_{B_ℓ} reply with $\nabla_{\theta_\ell} h|_{\hat{\theta}^{(k)}}$, computed according to (4.346).

e) The new estimate $\hat{\theta}^{(k+1)}$ is computed:

$$\begin{aligned}\hat{\theta}_\ell^{(k+1)} &= \hat{\theta}_\ell^{(k)} + \lambda_k \big(\nabla_{\theta_\ell} \log f_{A_\ell}|_{\hat{\theta}^{(k)}} \\ &\quad + \nabla_{\theta_\ell} \log f_{A_{\ell+1}}\big|_{\hat{\theta}^{(k)}} + \nabla_{\theta_\ell} h|_{\hat{\theta}^{(k)}} \big).\end{aligned} \tag{4.347}$$

f) Iterate 1–5.

As usual, several update schedules are possible. For example, in order to reduce the computational cost, one may prefer not to update the sum-product messages $\mu_{X_\ell \to f_{B_\ell}}(x_\ell)$ (cf. Step 3) at each iteration;

the probability functions $p_B(x_{\ell-1}, x_\ell, |y, \hat{\theta})$ (cf. Step 4) are then recomputed according to (4.344) using the *new* estimate $\hat{\theta}$, but the *old* messages $\mu_{X_\ell \to f_{B_\ell}}(x_\ell)$.

Remark 4.11. (Gradient EM vs. gradient sum-product)
If one replaces the messages $\nabla_{\theta_\ell}(\theta_k)$ (in Step 4 and 5) by the the sum-product messages $\nabla_{\theta_\ell}\mu_{f_{B_k}\to\Theta_k}(\theta_k)$ (or $\nabla_{\theta_\ell} \log \mu_{f_{B_k}\to\Theta_k}(\theta_k)$), one obtains a purely sum-product based gradient method (see Section 4.8.1).

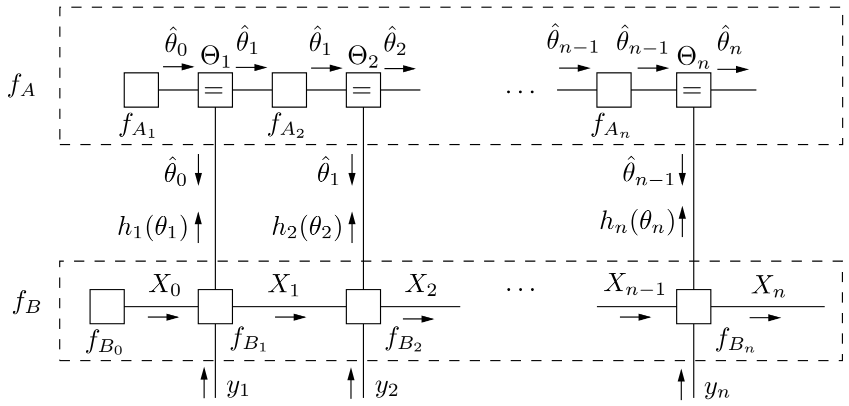


Figure 4.58: Gradient EM: stochastic approximation.

The stochastic approximation (SA) principle can also be applied to gradient EM, as illustrated in Fig. 4.58. It amounts to forward-only message-passing algorithms known as "recursive EM" or "online EM" [194] [207] [102] [226] [67]. In [194] [207], online EM algorithms for estimating *fixed* parameter are derived, whereas in [102] [226] [67], such algorithms are derived for *time-varying* parameters.

The example (4.339)–(4.340) can easily be extended to general functions f_A and f_B. The gradient of the h-message out of the generic node $g(z, \theta_m) \triangleq g(z_1, z_2, \ldots, z_n, \theta_m)$ (cf. Fig. 4.59) is as follows.

Gradient of an E-log message out of a generic node:

$$\nabla_{\theta_m} h(\theta_m) = \frac{\sum_z g(z, \hat{\theta}_m) \nabla_{\theta_m} \log g(z, \theta_m) \prod_{\ell=1}^{n} \mu_{Z_\ell \to \theta_m}(z_\ell)}{\sum_z g(z, \hat{\theta}_m) \prod_{\ell=1}^{n} \mu_{Z_\ell \to \theta_m}(z_\ell)}. \tag{4.348}$$

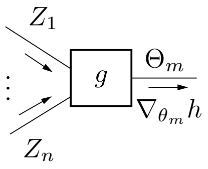


Figure 4.59: Generic node g.

We now state two interesting properties of gradient EM.

Theorem 4.6. Assume that a factor graph of a global function $f(x, \theta) \triangleq f_A(\theta) f_B(x, \theta)$ is available whose subgraph $f_B(x, \theta)$ is *cycle-free.* The fixed points of gradient EM applied on the graph of $f(x, \theta)$ are the stationary points of $f(\theta)$.

Proof: We will again use the function $\bar{f}(\theta, \theta')$ defined as

$$\bar{f}(\theta, \theta') \triangleq \sum_x f(x, \theta') \log f(x, \theta). \tag{4.349}$$

The fixed points of gradient EM are implicity defined as

$$\nabla_\theta \bar{f}(\theta, \theta^{\text{fixed}})\big|_{\text{fixed}} \overset{!}{=} 0. \tag{4.350}$$

From (4.168) it follows

$$\nabla_\theta f(\hat{\theta}^{\text{fixed}}) = 0, \tag{4.351}$$

and hence the fixed points of gradient EM are the stationary points (or "zero-gradient points") of $f(\theta)$. □

Theorem 4.7. Assume that a factor graph of a global function $f(x, \theta) \triangleq f_A(\theta) f_B(x, \theta)$ is available (whose subgraph $f_B(x, \theta)$ may be cycle-free or cyclic). The fixed points of a gradient EM algorithm applied on that factor graph are the stationary points of the function $\hat{f}(\theta)$, defined as:

$$\log \hat{f}(\theta) \stackrel{\triangle}{=} \log f_A(\theta) + \int_{-\infty}^{\theta} \mathrm{E}_{b(x|\tilde{\theta})}\left[\nabla_\theta \log f_B(x,\tilde{\theta})\right] d\tilde{\theta}, \tag{4.352}$$

where the beliefs $b(\cdot|\theta)$ are computed by means of the sum-product messages available at convergence of the sum-product algorithm.

The proof goes along the lines of the exposition in Example 4.5. Note that, if the subgraph f_B is cycle-free, then $\hat{f}(\theta) = f(\theta)$.

The Venn diagrams of Fig. 4.60 and Fig. 4.61 summarize our results concerning the fixed points of EM-type algorithms; Fig. 4.60 depicts the situation for factor graphs of $f(x,\theta)$ with *cycle-free* subgraphs $f_B(x,\theta)$, whereas Fig. 4.61 concerns *arbitrary* factor graphs of $f(x,\theta)$, i.e., factor graphs whose subgraph $f_B(x,\theta)$ may be *cycle-free* or *cyclic.*

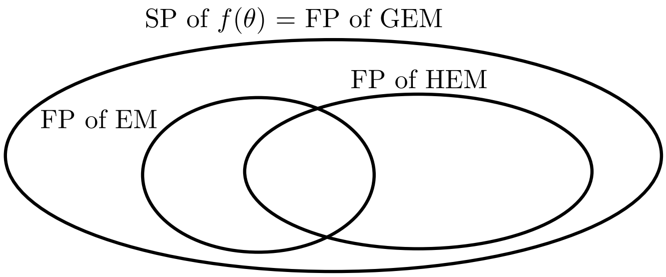


Figure 4.60: Venn diagram depicting the stationary points (SP) of $f(\theta)$, fixed points (FP) of EM, hybrid EM (HEM), and gradient EM (GEM); for *cycle-free* subgraph $f_B(x,\theta)$.

4.9.7 Application

In the constant-phase model, the maximization step (4.177) can be carried out analytically; in the random walk phase model, the maximization step is intractable and we will apply steepest descent, resulting in a gradient-EM algorithm. First we treat the constant-phase model, then the random-walk phase model.

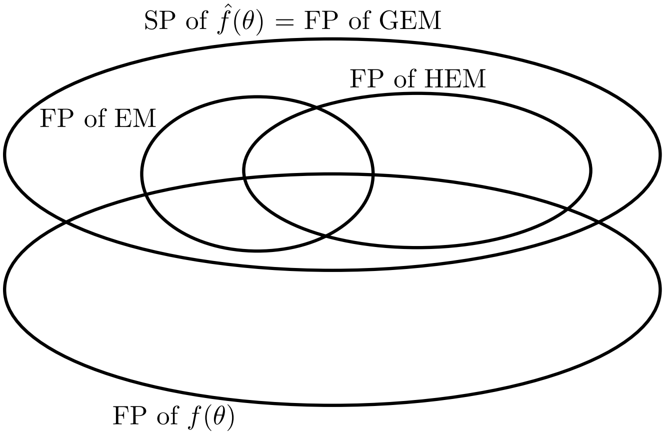


Figure 4.61: Venn diagram depicting the stationary points (SP) of $f(\theta)$ and $\hat{f}(\theta)$, fixed points (FP) of EM, hybrid EM (HEM), and gradient EM (GEM); for *cycle-free* and *cyclic* subgraphs $f_B(x, \theta)$.

Constant-phase model

We start by choosing the "boxes" f_A and f_B appropriately. Since we wish to estimate the phase Θ, we choose the Θ edges as cut set of edges that separates the boxes f_A and f_B (see Fig. 4.4 and Fig. 4.62). The box f_A contains all nodes that correspond to the *prior* of Θ, i.e., all nodes that are solely connected to Θ edges. In this case, f_A exclusively contains the equality constraint node Θ, as illustrated in Fig. 4.62; all other nodes belong to the box f_B. Note that the box f_B has a "nice" structure, i.e., f_B has a non-trivial factorization. We now apply the message-update rules (4.174) and (4.177) to the factor graph of Fig. 4.62. The message $h(\theta)$ equals

$$h(\theta) = \sum_{k=1}^{L} \sum_{x_k} p_B(x_k | y, \hat{\theta}^{(\ell)}) \log f_B(x_k, y_k, \theta) \tag{4.353}$$

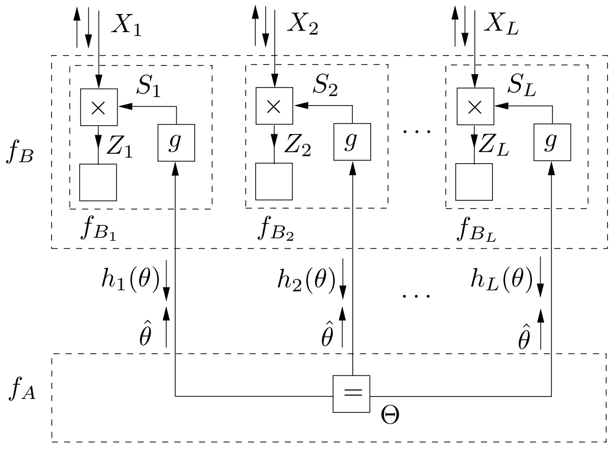


Figure 4.62: EM in the constant-phase model.

$$\triangleq \sum_{k=1}^{L} h_k(\theta), \tag{4.354}$$

where the function $f_B(x_k, y_k, \theta)$ is given by

$$f_B(x_k, y_k, \theta) \triangleq \int_{z_k} \oint_{s_k} \delta(z_k - x_k s_k)\delta(s_k - e^{j\theta}) \cdot$$
$$(2\pi\sigma_N^2)^{-1}\, e^{-|y_k - z_k|^2/2\sigma_N^2} dz_k ds_k \tag{4.355}$$
$$= (2\pi\sigma_N^2)^{-1}\, e^{-|y_k - x_k e^{j\theta}|^2/2\sigma_N^2}. \tag{4.356}$$

The small dashed boxes in Fig. 4.62 represent the functions $f_B(x_k, y_k, \theta)$. The marginalization (4.355)–(4.356) corresponds to "closing" those boxes.

The distribution $p_B(x_k|y, \hat{\theta}^{(\ell)})$ is defined as

$$p_B(x_k|y, \hat{\theta}^{(\ell)}) \triangleq \gamma_k f_B(x_k, y_k, \hat{\theta}^{(\ell)}) \mu_{X_k \to \boxtimes}(x_k), \tag{4.357}$$

where γ_k is a normalization factor, i.e.,

$$\gamma_k \triangleq \left(\sum_{x_k} f_B(x_k, y_k, \hat{\theta}^{(\ell)}) \mu_{X_k \to \boxtimes}(x_k) \right)^{-1}. \tag{4.358}$$

Note that $f_B(x_k, y_k, \hat{\theta}^{(\ell)})$ is nothing but the message $\mu_{\boxtimes \to X_k}(x_k)$, which leaves the multiply node along the edge X_k (at the ℓ-th iteration). Therefore

$$p_B(x_k|y, \hat{\theta}^{(\ell)}) \stackrel{\triangle}{=} \gamma_k \mu_{X_k \to \boxtimes}(x_k) \mu_{\boxtimes \to X_k}(x_k). \tag{4.359}$$

In the following, we use the short-hand notation $p_B(x_k)$ for $p_B(x_k|y, \hat{\theta}^{(\ell)})$. The phase Θ has a uniform prior, i.e., $f_A(\theta) \stackrel{\triangle}{=} 1$ for all θ. The message $\hat{\theta}^{(\ell+1)}$ equals

$$\begin{aligned}
\hat{\theta}^{(\ell+1)} &= \operatorname*{argmax}_{\theta} h(\theta) && (4.360)\\
&= \operatorname*{argmax}_{\theta} \sum_{k=1}^{L} \sum_{x_k} p_B(x_k) \log f_B(x_k, y_k, \theta) && (4.361)\\
&= \arg \sum_{k=1}^{L} \sum_{x_k} p_B(x_k)\, [y_k x_k^*] && (4.362)\\
&= \arg \sum_{k=1}^{L} \left[y_k \left(\mathrm{E}[x_k]\right)^* \right], && (4.363)
\end{aligned}$$

where the expectation $\mathrm{E}[x_k]$ is defined as

$$\mathrm{E}[x_k] \stackrel{\triangle}{=} \sum_{x_k} p_B(x_k)\, x_k. \tag{4.364}$$

In this particular case, a closed-form expression for the messages $\hat{\theta}$ can thus be found. The expression (4.363) was proposed earlier by Noels et al. [185].

Random-walk phase model

We again start by determining the boxes f_A and f_B; the box f_B is chosen as in the constant-phase model. The box f_A is now more interesting, as illustrated in Fig. 4.63: it contains the nodes $p(\theta_k|\theta_{k-1})$ besides the equality constraint nodes Θ_k. Both f_A and f_B have in this case a non-trivial structure. We now apply the message-update rules (4.174) and (4.177) to the factor graph of Fig. 4.63. The message $h(\theta)$, summarizing the box f_B, is again given by (4.353), since the box f_B remained unchanged.

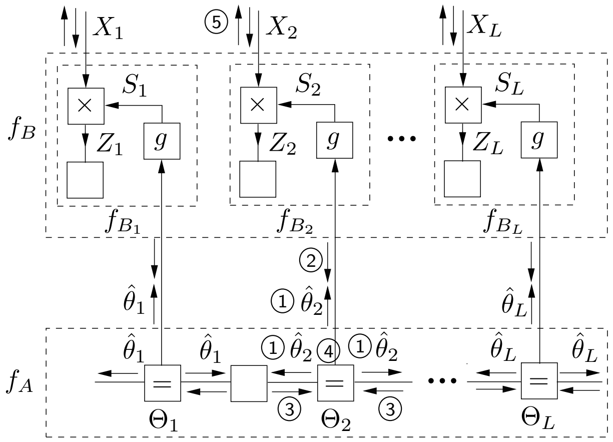


Figure 4.63: Steepest descent-EM in the random-walk phase model.

The function $\log f_A(\theta)$ equals

$$\log f_A(\theta) \quad = \quad \sum_{k=2}^{L} \log p(\theta_k|\theta_{k-1}), \tag{4.365}$$

where

$$p(\theta_k|\theta_{k-1}) \triangleq (2\pi\sigma_W^2)^{-1/2} \sum_{n\in\mathbf{Z}} e^{-(\theta_k-\theta_{k-1}+n2\pi)^2/2\sigma_W^2}. \tag{4.366}$$

The message $\hat{\theta}$ is computed as

$$\hat{\theta} = \operatorname*{argmax}_{\theta}(\log f_A(\theta) + h(\theta)) \tag{4.367}$$

$$= \operatorname*{argmax}_{\theta}\Bigg[\sum_{k=2}^{L} \log p(\theta_k|\theta_{k-1}) + \sum_{k=1}^{L}\sum_{x_k} p_B(x_k)\log f_B(x_k,y_k,\theta_k)\Bigg]. \tag{4.368}$$

The maximization (4.368) can not be carried out analytically. We solve this problem by gradient EM. We propose two approaches; the first algorithm is derived by straightforwardly applying the generic rules, the second method is an SA algorithm.

The first method performs following steps (see Fig. 4.63):

① The equality constraint nodes Θ_k broadcast the estimates $\hat{\theta}_k^{(\ell)}$.

② The nodes f_{B_k} reply with $\frac{dh_k(\theta_k)}{d\theta_k}\Big|_{\hat{\theta}_k^{(\ell)}}$.

③ The nodes $p(\theta_k|\theta_{k-1})$ and $p(\theta_{k+1}|\theta_k)$ reply with $\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_k}\Big|_{\hat{\theta}^{(\ell)}}$ and $\frac{\partial \log p(\theta_{k+1}|\theta_k)}{\partial\theta_k}\Big|_{\hat{\theta}^{(\ell)}}$ respectively.

④ The new estimate $\hat{\theta}^{(\ell+1)}$ is computed:

$$\hat{\theta}_k^{(\ell+1)} = \hat{\theta}_k^{(\ell)} + \lambda\Bigg(\frac{\partial \log p(\theta_k|\theta_{k-1})}{\partial\theta_k}\Bigg|_{\hat{\theta}_k^{(\ell)}} + \frac{\partial \log p(\theta_{k+1}|\theta_k)}{\partial\theta_k}\Bigg|_{\hat{\theta}_k^{(\ell)}} + \frac{dh_k(\theta_k)}{d\theta_k}\Bigg|_{\hat{\theta}_k^{(\ell)}}\Bigg). \tag{4.369}$$

⑤ The messages $\mu_{\boxtimes\to X_k}(x_k)$ are updated.

The steps ①–④ are iterated a number of times before ⑤ is carried out. Note that the above algorithm (Step 1–6) is very similar to the steepest-descent algorithm of Section 4.8.4. There, steepest descent is applied to sum-product messages, whereas it is applied to the E-log messages h_k here. If one replaces h_k by $\log\mu_{g\to\Theta_k}$ in the above procedure (in Step ② and ④), one obtains the (first) gradient-based sum-product algorithm of Section 4.8.4.

One obtains an SA EM algorithm by replacing the messages $\log\mu_{g\to\Theta_k}$ in the SA gradient algorithm of Section 4.8.4 by the E-log messages h_k (in Step ① and ③). The resulting SA EM algorithm is similar to the algorithm developed in parallel work by Noels et al. [151].

4.9.8 Summary

The formulation of the EM algorithm as message passing on a graph has some interesting implications:

- A **local view** onto the EM algorithm has been given. It is not necessary to handle complicated equations for the complete model. The problem is divided into simple and small units. The global model is built by connecting these units into a graph. The EM algorithm is then computed by passing messages along the edges of this graph.

- A **generic update rule** for EM messages on a factor graph has been given, i.e., the E-log rule (4.226). Given the node function and the incoming sum-product messages this rule leads to the message sent along the edges modeling the parameters.

- A **table of ready-to-use nodes** has been given (Table 4.2). This table was derived by applying the update rule (4.226) to nodes that often occur in signal-processing applications.

- The message passing EM **fits well into the factor graph framework** developed so far. Once a probabilistic problem is modeled as a factor graph, different algorithms can be used by passing different messages along the edges of the graph. The EM messages are an alternative among others.

- There is much **flexibility in choosing a schedule** which opens up the opportunity to develop different forms of the EM algorithm, especially online estimation algorithms.

- It is also possible to **combine with other message types**. Either the E-step as well as the M-step can apply different techniques such as simulation-based methods (**stochastic EM**, **Monte-Carlo EM**), gradient methods (**gradient EM**), and decision-based methods (**AECM**).

- **Non-trivial a priori models** for the parameters are possible. The maximization in the E-step amounts to the max-product algorithm on the graph for the parameters.

- A **hybrid update rule** has been devised, i.e., the hybrid E-rule (4.261). The fixed points of algorithms that use the hybrid E-rule are stationary points of the marginal $f(\theta)$, as in standard EM. The hybrid rule sometimes leads to simpler expressions. It is also be applied to deterministic nodes, in contrast to the E-log rule (4.226).

- The E-log rule and hybrid E-rule can be applied on **cyclic subgraphs** $f_B(x, \theta)$. We characterized the fixed points of the resulting (hybrid) EM-algorithms.

4.10 Results and Discussion

We performed simulations of the proposed code-aided phase-estimation algorithms for the constant-phase model and random-walk phase model. In particular, we assessed the performance of the phase estimators based on:

- numerical integration (NI) (cf. Section 4.5.2)
- particle methods (PM) (cf. Section 4.6.7)
- adaptive quantization (AQ) (cf. Section 4.7.2)
- sum-product based steepest descent (SP-SD) (cf. Section 4.8.4)
- EM (constant-phase model) and gradient EM (GEM) (random-walk phase model) (cf. Section 4.9.7).

We used a rate 1/2 LDPC code of length $L = 100$ that was randomly generated and was not optimized for the channel at hand. The factor graph of the code does not contain cycles of length four. The degree of all bit nodes equals 3; the degrees of the check nodes are distributed as follows: 1, 14, 69 and 16 check nodes have degree 4, 5, 6 and 7 respectively. The symbol constellation was Gray-encoded 4-PSK. We iterated 20 times between the LDPC decoder and the phase estimator, each time with hundred iterations inside the LDPC decoder. In the gradient-based algorithms and the adaptive-quantization-based algorithms, we iterated 50 times inside the factor graph of the phase model. We did not iterate between the LDPC decoder and the mapper. We optimized the (constant) step size parameter λ which occurs in the gradient-based algorithms. We considered the phase noise values $\sigma_W^2 = 0$ (constant-phase model) and $\sigma_W^2 = 10^{-4}$ rad^2 (random-walk phase model).

We are fully aware of the fact that this setup is rather *artificial*. First of all, one normally uses longer codes (e.g., $L = 10.000$). Since some of the phase estimators are complex, however, particularly the estimators based on (adaptive) quantization, it is convenient to use short block codes in order to limit the required computation time.

In addition, the code we used is randomly generated and is therefore not optimized for the channel at hand. It is clear that the channel capacity can not be achieved by means of this code, which is also not our aim; we mainly wish to *compare* the performance of our code-aided phase estimators.

Moreover, the phase noise may in some applications be stronger than $\sigma_W^2 = 10^{-4}$ rad^2. Pilot symbols seem then to be required. The issue of pilot sequence design, however, goes beyond the scope of this thesis; therefore, in order to avoid the use of pilot symbols, we only consider channels with weak phase noise. For simplicity, we will also assume that the phase ambiguity has been resolved.[12]

In the particle-based algorithms, the messages were represented as lists of $N = 200$ particles. In the numerical-integration based algorithms, the phase was uniformly quantized over $N = 200$ levels; in the algorithms based on adaptive quantization, we used $N = 200$ (non-uniform) quantization levels. In the following, we elaborate on the performance, robustness, convergence, and complexity of the proposed code-aided phase estimators.

4.10.1 Performance

In Fig. 4.64 and Fig. 4.65, the phase synchronizers for the constant-phase model and random-walk phase model ($\sigma_W^2 = 10^{-4}$ rad^2) respectively are compared in terms of the mean squared (phase) estimation error (MSE); in Fig. 4.66 and Fig. 4.67, the phase synchronizers for the constant-phase model and random-walk phase model ($\sigma_W^2 = 10^{-4}$ rad^2) respectively are compared in terms of the (decoded) frame error rate (FER).

From Fig. 4.64 and Fig. 4.65 it becomes clear that the phase estima-

[12]In [52] we propose techniques to resolve the phase ambiguity (see also [216]). In the constant-phase model, the phase ambiguity can be resolved by testing the possible hypothesis and by choosing the most probable hypothesis. This method can also be applied to the random-walk phase model as long as the phase noise is sufficiently small ($\sigma_W^2 \leq 10^{-4}$ rad^2); otherwise, one may resort to pilot symbols.

tors based on adaptive quantization (AQ) have the smallest MSE, followed by the phase estimators based on numerical integration (NI), the particle methods (PM)), the EM-based algorithm (constant phase), the phase estimator gradient EM (GEM) (random-walk phase), and the sum-product-based steepest descent (SP-SD) estimators. As can be seen from Fig. 4.66 and Fig. 4.67, the FER of all code-aided phase synchronizers is about the same; they all perform significantly better than the classical M-law estimator, which does not use any information from the decoder [136]. The code-aided phase estimators based on adaptive quantization and uniform quantization have a slightly smaller FER than the other code-aided phase estimators.

Simulation results (not shown here) indicate that the performance of GEM-based and SP-SD-based phase estimators does not depend on the message-update schedule inside the factor graph of the phase model: the standard schedule and the stochastic approximation (SA) approach (cf. Section 4.8.3) lead to about same performance. In Fig. 4.64 to Fig. 4.67, the results for the SA-approach are shown; the curves for the standard schedule practically coincide (not shown).

We investigated why the EM-based algorithm and the gradient-based algorithms perform (slightly) worse in terms of MSE and FER than the numerical-integration-based and adaptive-quantization-based approach. The performance degradation is mainly due to two factors. First of all, the EM-based algorithm and the gradient-based algorithms approximate a density ("message") by a single value. We quantified the resulting FER degradation by considering a slight modification of the NI-based algorithm for the random-walk phase model, where the upward Θ_k-messages (arriving at the multiply nodes) are not represented as quantized messages (as in the NI-based algorithm), but as single values, i.e., as the mode of the upward Θ_k-messages; all other Θ_k-messages are quantized messages (as in the NI-based algorithm). The FER gap between this modified NI-based algorithm and the (unmodified) NI-based algorithm is *solely* due to the single-value approximation of the upward Θ_k-messages. The FER of the modified NI-based algorithm is shown in Fig. 4.68 together with the FER of the SP-SD-based and NI-based algorithm. For SNR-values smaller or equal to 2dB, the FER of the modified NI-based algorithm coincides with the FER of the SP-SD algorithm. At higher SNR values, it coincides with the FER of the (unmodified) NI-based algorithm. This is in agreement with our intuition: at high SNR, the width of the posterior density $p(\theta|y)$ is small and the Dirac-delta approximation ("single value") is satisfactory; this is not the case at low SNR. From Fig. 4.68 we learn that there must be *additional* factors that lead to performance degradation for the SP-SD-based and EM-based estimators, since at high SNR, the Dirac delta approximation is satisfactory and does not lead to FER degradation. A second factor is the fact that both estimators do sometimes not convergence to the global maximum of the marginal $p(\theta|y)$, but to a neighboring local maximum. Fig. 4.69 shows histograms of phase estimates $\hat{\theta}$ obtained by the NI-based, AQ-based, SP-SD-based, and EM-based phase estimator for the constant-phase model at SNR = 3dB, where the true value $\Theta = 0.5$ rad. We resolved the phase ambiguity by limiting the phase estimates to the interval $[0, \pi/2)$. Note that the histograms of the EM-based and SP-SD-estimators contain a significant number of outliers ($\hat{\theta} = 0$ and $\hat{\theta} = \pi/2$), in contrast to the histograms of the other two estimators. The outliers most often occur when the initial estimate (generated by the M-law) is closer to a local maximum of the marginal $p(\theta|y)$ than to the global maximum, as can be seen from Fig. 4.70. This problem can often be alleviated by running the (SP-SD-based or EM-based) phase estimator several times, each time with a different initial estimate $\hat{\theta}^{(0)}$. Wymeersch [216] has improved the performance of the EM-based phase estimator for the constant-phase model by using multiple initial estimates $\hat{\theta}^{(0)}$.

4.10.2 Convergence

Fig. 4.71 and Fig. 4.72 depicts the FER of the EM-based phase estimator and the NI-based phase estimator respectively as a function of the number of iterations (for $\sigma_W^2 = 0$ and 10^{-4} rad^2). The curves show that 10 iterations between the LDPC decoder and the phase estimators are sufficient to achieve convergence. We verified that this statement also holds for the other algorithms (results not shown here).

4.10.3 Robustness

We investigated how the performance of our algorithms depends on some of their parameters, i.e., the number of quantization levels or particles, and the step size λ. We also verified how the performance degrades when the (true) value for σ_N is unknown to the receiver and a wrong

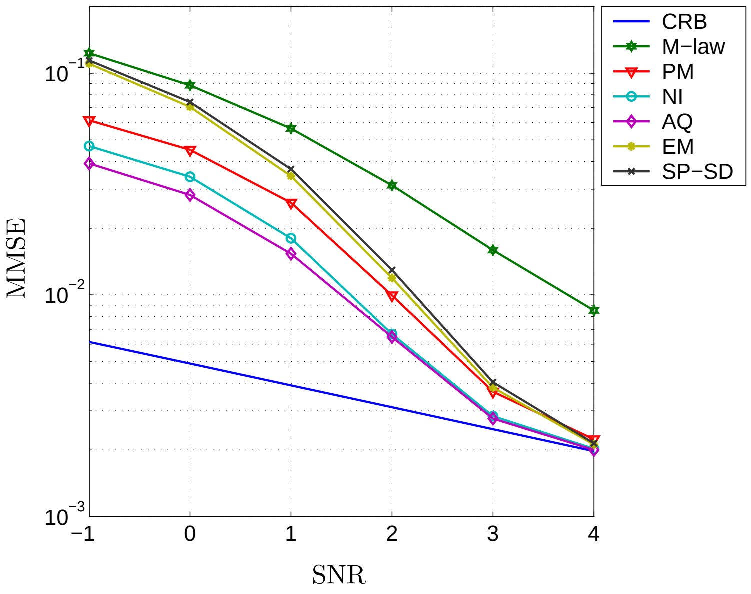


Figure 4.64: MSE for the constant-phase model.

value $\sigma_R \neq \sigma_N$ is inserted in (4.7).

Fig. 4.73 shows how the FER of the NI-based phase estimator (for the random-walk phase model) varies with the number of quantization levels; it can be seen from this figure that 100 quantization levels are sufficient. We verified that the same holds for the AQ-based phase estimator, and that in the particle methods, 100 particles suffice (results not shown here).

In Fig. 4.74, the FER of the standard SP-SD estimator (for the random-walk phase model) is shown for various values of the learning rate λ (cf. (4.132)). It can be seen that the optimum learning rate depends only weakly on the SNR. We obtained similar curves for the other gradient-based phase estimators (not shown here).

In Fig. 4.75, we consider the situation where the receiver uses the wrong value of σ_N, i.e., in (4.7) σ_N is replaced by a value $\sigma_R \neq \sigma_N$. Fig. 4.75 illustrates how the MSE of the NI-based phase estimator (for the constant-

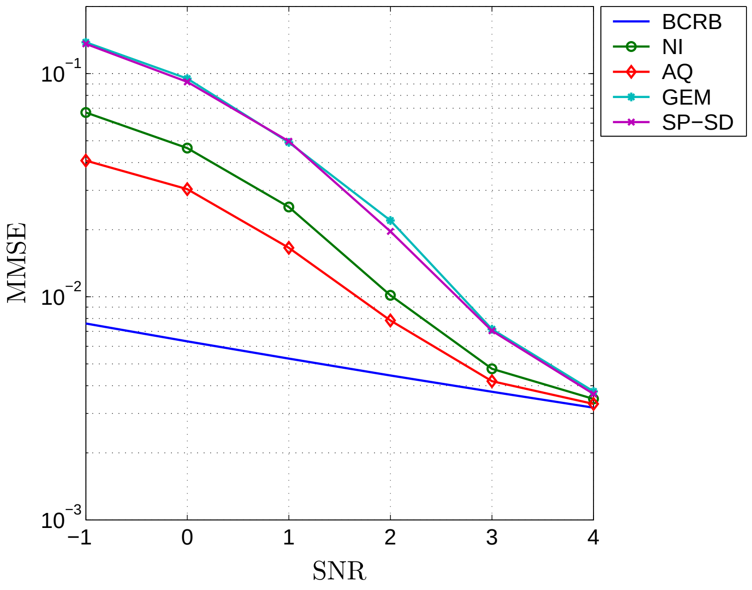


Figure 4.65: MSE for the random-walk phase model with $\sigma_W^2 = 10^{-4}$ rad^2.

phase model) depends on the value σ_R. As expected, the minimum MSE is achieved when there is no mismatch, i.e., for $\sigma_R = \sigma_N$.

4.10.4 Complexity

The EM-based phase estimator and the gradient-based phase estimators have the lowest complexity, since the messages are represented by a single value. The algorithms based on numerical integration are much more complex. As is well known, numerical integration becomes infeasible in high-dimensional systems. The particle methods are complex as well, but they scale better with the dimensionality of the system. The same holds for the approach based on adaptive quantization. In the following section, we summarize the main points of this chapter.

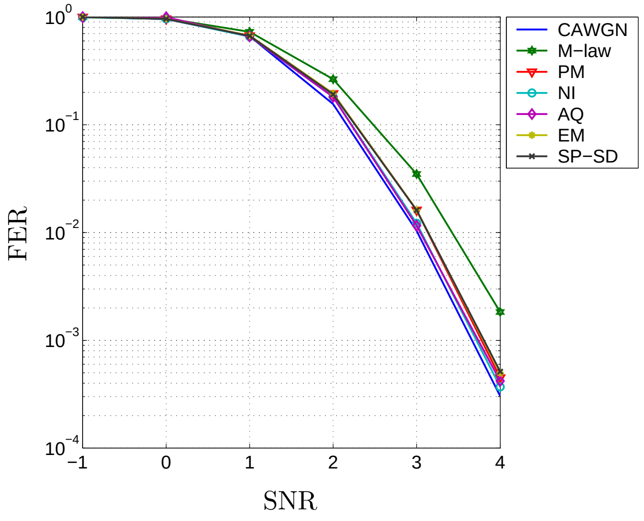


Figure 4.66: FER for the constant-phase model.

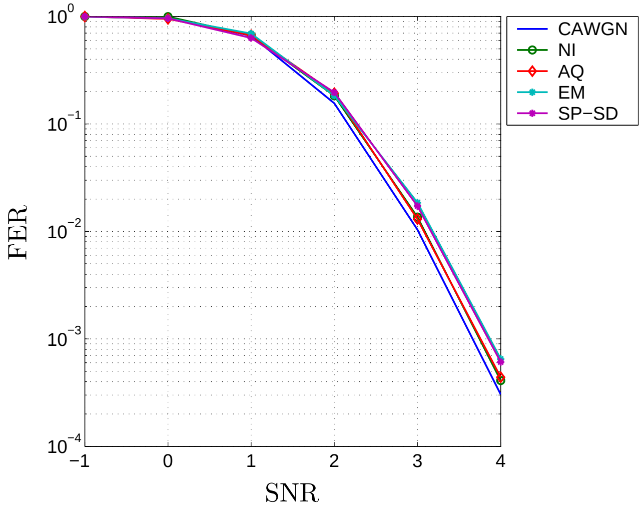


Figure 4.67: FER for the random-walk phase model with $\sigma_W^2 = 10^{-4}$ rad^2.

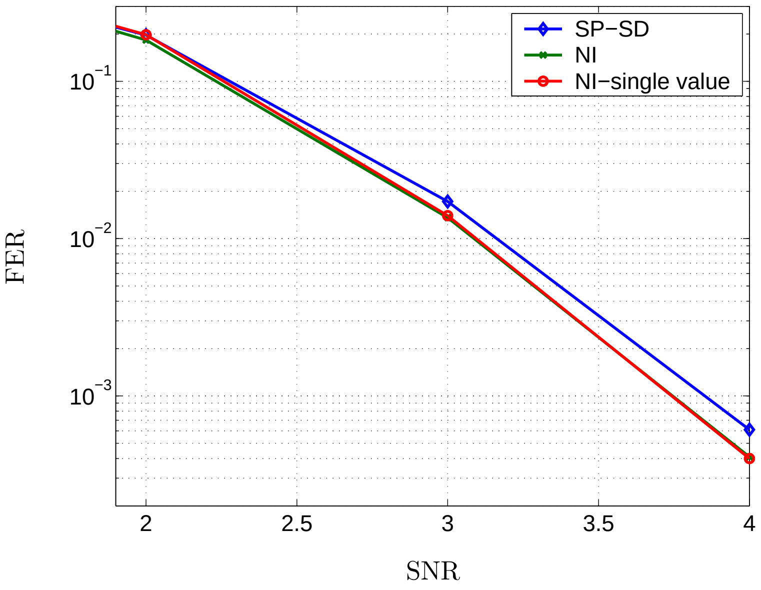
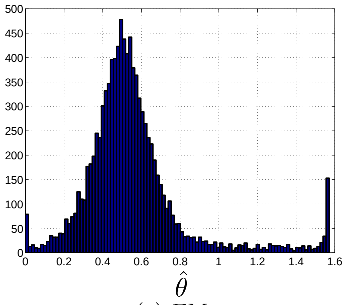
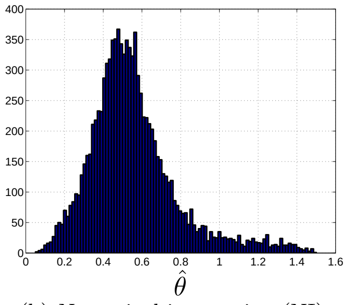
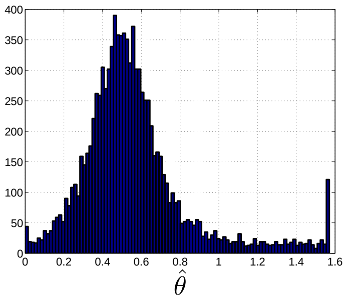


Figure 4.68: Frame error rate for the random-walk phase model with $\sigma_W^2 = 10^{-4}$ rad^2 for the SP-SD-based algorithm, the NI-based algorithm and the modified NI-based algorithm, where the upward Θ_k-messages are represented by a single value.


(a) EM.


(b) Numerical integration (NI).


(c) Sum-product based steepest descent (SP-SD).

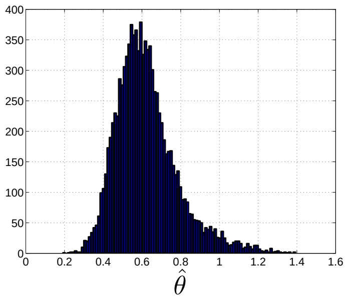

(d) Adaptive quantization (AQ).

Figure 4.69: Histograms of the phase estimates $\hat{\theta}$ for the constant-phase model (SNR = 0dB); the true value is $\theta = 0.5$ rad.

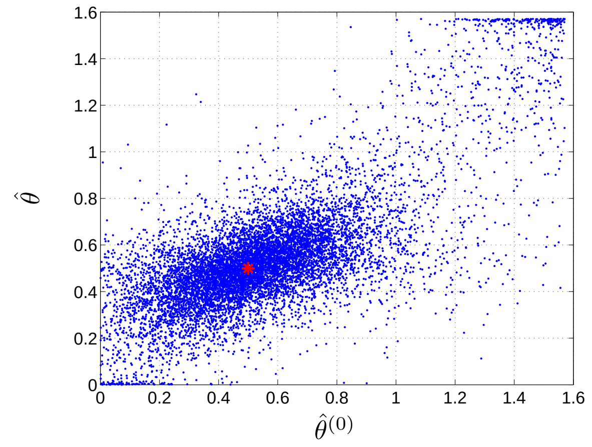


Figure 4.70: Initial estimate $\hat{\theta}^{(0)}$ (obtained by the M-law) vs. (final) estimate $\hat{\theta}$ obtained after 20 iterations of the EM-based phase estimator ($\sigma_W^2 = 0$ rad^2; SNR = 0dB). The true value $\theta = 0.5$ rad is depicted by the star centered at (0.5,0.5).

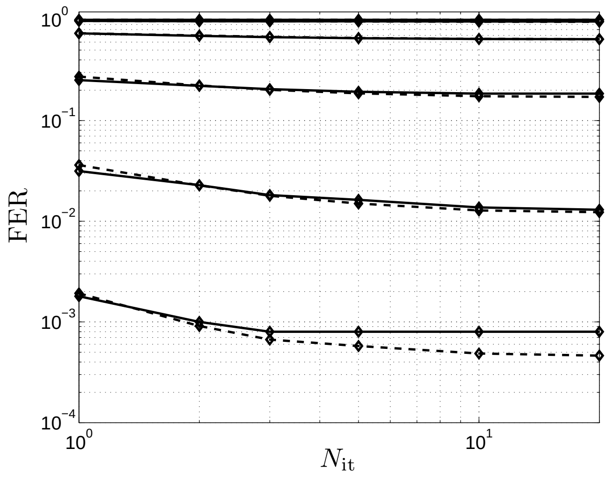


Figure 4.71: FER of EM-based algorithm as a function of the iteration number ($\sigma_W^2 = 0$ and 10^{-4} rad^2; SNR = -1, 0, . . . , 4dB).

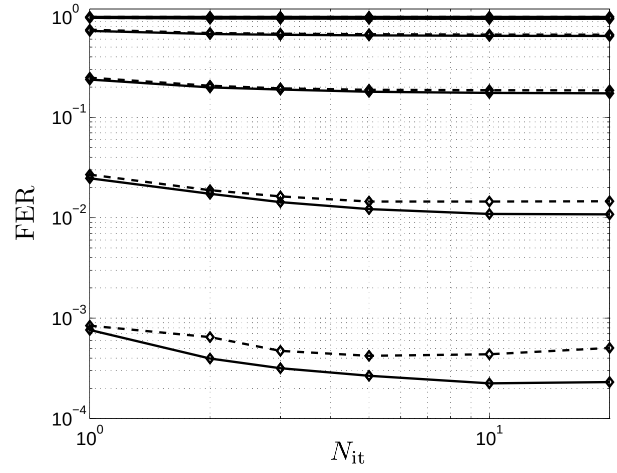


Figure 4.72: FER of NI-based approach as a function of the iteration number ($\sigma_W^2 = 0$ and 10^{-4} rad^2; SNR = -1, 0, ..., 4dB).

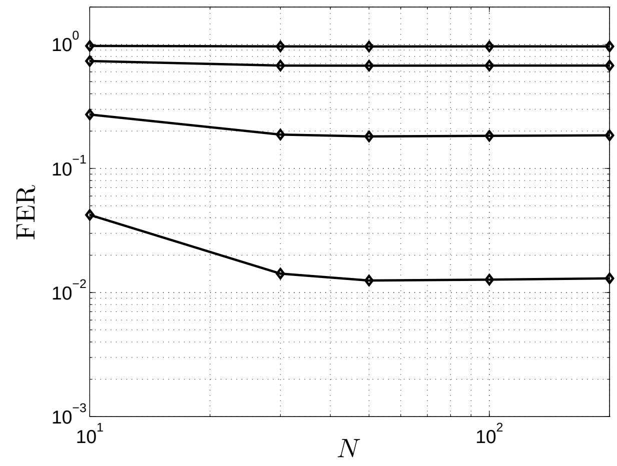


Figure 4.73: FER of the NI-based estimator as a function of the number of quantization levels N ($\sigma_W^2 = 10^{-4}$ rad^2; SNR = -1, 0, ..., 3dB).

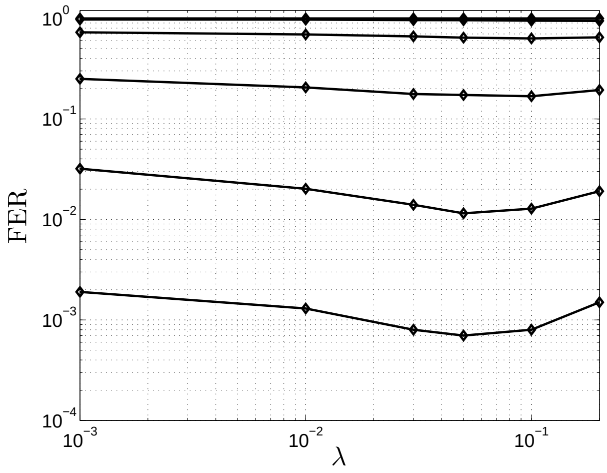


Figure 4.74: FER as a function of the step size λ ($\sigma_W = 10^{-4}$ rad^2; SNR = -1, 0, ..., 4dB).

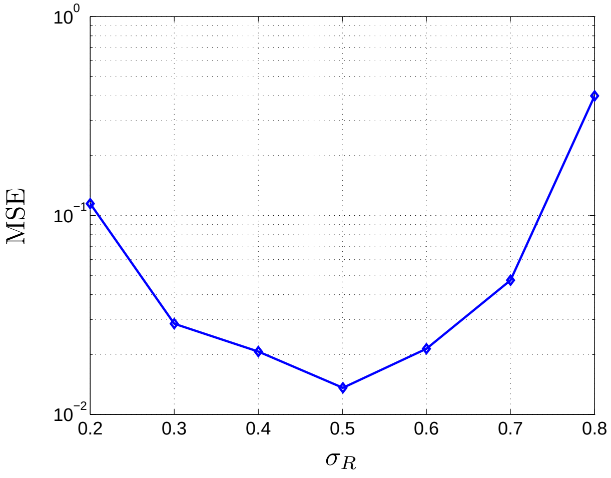


Figure 4.75: MSE as a function of the value σ_R ($\sigma_W^2 = 0$ rad^2; $\sigma_N = 0.5006$; SNR = 3dB).

4.11 Summary

In this chapter, we described how factor graphs can be used for statistical inference, i.e., detection and estimation. Statistical inference is accomplished by sending messages along the edges of the graph ("summary propagation" or "message passing"). Different algorithms are obtained by different message types or different message-update schedules.

We described various standard algorithms in signal processing and machine learning as message passing on factor graphs:

- particle methods, e.g., Gibbs sampling, particle filtering, importance sampling, simulated annealing, Markov-Chain Monte-Carlo methods
- gradient-based methods, e.g., steepest ascent/descent
- expectation maximization (EM) and extensions, e.g., Monte-Carlo EM, gradient EM, SAGE, etc.
- decision-based methods (e.g., iterative conditional modes).

We determined the local message-update rules for each of the above algorithms. Those update rules may be used as building blocks for novel estimation and detection algorithms; by listing the possible update rules at each node in the factor graph, one can *systematically* derive novel algorithms. We derived various code-aided phase-estimation algorithms in this fashion.

Chapter 5

Computing Cramér-Rao-Type Bounds

In this chapter, we present message-passing algorithms to compute Cramér-Rao-type bounds, which are lower bounds on the mininum mean squared error. Cramér-Rao-type bounds can be used to asses the performance of estimation algorithms, in particular, code-aided phase-estimation algorithms (see Example 5.5, Example 5.6, and Example 5.11). The results of this chapter are based on [40] and [41].

5.1 Introduction

For many practical estimation problems[1] (e.g., code-aided carrier-phase estimation), popular estimators such as the maximum likelihood estimator (ML), the maximum a posteriori estimator (MAP) or the minimum mean square error estimator (MMSE) are infeasible. One therefore often resorts to approximate methods such as expectation maximization [58], loopy belief propagation [119], gradient-based algorithms [19], Markov

[1]Basic notions from estimation and detection theory are reviewed in Appendix A.

Chain Monte Carlo methods [171] (MCMC), particle filters [59], or combinations of those methods.

Suboptimal estimators are typically compared based on their mean squared estimation error (MSE). However, the MSE is not an *absolute* performance measure; in order to determine whether a suboptimal algorithm is close to optimal (in terms of MSE), the MSE of the minimum mean squared error (MMSE) estimator is required. Unfortunately, the mimimum achievable MSE can often not be computed (neither analytically, nor numerically), and one needs to resort to *bounds* on the mimimum achievable MSE, typically, *lower* bounds. A well-known family of such lower bounds are the Cramér-Rao-type bounds. In this chapter, we present (novel) algorithms for computing Cramér-Rao-type bounds for real-life estimation problems. Interestingly, Cramér-Rao-type bounds are tight for many (practical) estimation problems.

For the estimation of *parameters*, a commonly used lower bound for the MSE is the Cramér-Rao bound (CRB), given by the inverse of the Fisher information matrix [199] [176] ("standard CRB"). The CRB has been computed in a wide variety of contexts, ranging from communications (e.g., [16] [206] [138] [76] [86] [183] [184] [228] [145] [146] [147] [149] [150] [18] [227]), signal and image processing (e.g., [71] [195] [111] [29] [107] [142] [162] [70] [163] [14]), to computer vision (e.g., [221] [159]). For some applications, a closed-form expression for the CRB is available; in other applications, e.g., estimation in AR(MA)-models [71] [195] [111] [29] [107] [142] [162] [70] [163], the derivation of CRBs is involved. For example, the CRB has been derived for AR(MA)-models *without* observation noise [71] [195] [111] [29] [107] [142] [162], but for AR(MA)-models *with* observation noise, the CRB seems to be intractable [163]; therefore, one often resorts to asymptotic bounds (i.e., high-SNR bounds) [70] or to numerical algorithms [163].[2]

Van Trees derived an analogous bound to the CRB for *random variables*, referred to as "Bayesian CRB" (BCRB) or "posterior CRB" or "Van Trees bound" [199]. Rather surprisingly, far less attention has been given to the BCRB than to the standard CRB. The BCRB has been determined for a few estimation problems; Tichavský et al. derived the BCRB for filtering in state-space models with *freely* evolving state [192]. A particle method for computing the BCRB of [192] for the particular case of non-

[2]The algorithm of [163] *only* applies to ARMA models and is not easily extended to other systems.

linear non-stationary dynamical systems is presented in [191]; the method of [191] has recently been used for computing the BCRB for various tracking problems (see e.g., [26]).

Recently, so-called *hybrid* Cramér-Rao bounds have been proposed [172]; they apply to the joint estimation of parameters and random variables. In this chapter, we consider each of the three different types of Cramér-Rao bounds, i.e., standard, Bayesian, and hybrid Cramér-Rao bounds.

There are two general strategies to obtain Cramér-Rao-type bounds for a given estimation problem. One may derive Cramér-Rao-type bounds from the information matrix of the *joint* probability density function (pdf) of the system at hand; alternatively, one may derive such bounds from information matrices of *marginals* of the joint pdf. In this chapter, we propose (novel) practical algorithms to compute Cramér-Rao bounds:

- following each of both strategies,
- for each of the three different types of Cramér-Rao bounds.

Our algorithms are message-passing algorithms that operate on a factor graph of the system at hand. The algorithms can be applied to standard estimation problems, such as:

- filtering and smoothing in state-space models,
- estimation of the parameters of state-space models,
- estimation in multiple coupled state-space models and other systems that are most naturally represented by cyclic graphs,
- code-aided channel estimation.

The first three problems are ubiquitous in various areas of signal processing such as biomedical signal processing, speech and image processing; the last problem appears in the context of communications. Our algorithms sometimes lead to analytical bounds; for most non-trivial estimation problems, however, the bounds involve *intractable* integrals, which we then solve by Monte-Carlo integration.

In this chapter, we will give numerous examples, ranging from toy examples that illustrate the (sometimes abstract) concepts to more challenging problems such as carrier-phase estimation and estimation in AR models.

Our algorithms for computing Cramér-Rao-type bounds may also lead to novel estimation algorithms, more precisely, to estimation algorithms that are based on the *natural gradient*, which is a central concept in information geometry [7]. More generally speaking, our algorithms open the door to promising (practical) applications of information geometry to (estimation with) graphical models.

We organized this chapter as follows. In the next section, we review the three Cramér-Rao-type bounds, and outline the two general strategies to compute Cramér-Rao-type bounds. In Section 5.3, we present message-passing algorithms for computing Cramér-Rao-type bounds from information matrices of *joint* pdfs; we illustrate the techniques by several standard estimation problems, e.g., estimation in (general) state-space models, carrier-phase estimation, and estimation of the parameters and state of AR models. In Section 5.4, we propose algorithms to compute Cramér-Rao-type bounds from information matrices of *marginals.* We apply also these methods to estimation in (general) state-space models and to estimation in AR models; we elaborate on code-aided channel estimation. In Section 5.5, we summarize our methods and contributions; in Section 5.6, we outline several extensions of our methods, notably, natural-gradient based algorithms (and information geometry in general), other types of bounds, and other types of error measures.

The proofs of the lemmas and theorems of this chapter can be found in Appendix K, unless stated otherwise.

5.2 Overview of Cramér-Rao-Type Bounds

We review:

a) the (standard) Cramér-Rao bound, which applies to parameters,

b) Bayesian Cramér-Rao bounds, which apply to random variables,

c) hybrid Cramér-Rao bounds, which are applicable to the joint estimation of parameters and random variables.

5.2.1 Standard Cramér-Rao Bound

We start by introducing our notation. Let $\Theta = (\Theta_1, \Theta_2, \ldots, \Theta_n)^T$ be a parameter vector, and let $Y = (Y_1, Y_2, \ldots, Y_N)^T$ be a real random vector (the extension to complex random vectors is straightforward). Suppose that $p(y|\theta)$ is the probability density function (pdf) of Y, which is parametrized by Θ. We consider the problem of estimating Θ from an observation vector $y = (y_1, y_2, \ldots, y_N)^T$. Let the function $\hat{\theta}(y)$ be an estimator of Θ based on the observation y. We define the error matrix $\mathbf{E}(\theta)$ as:

$$\mathbf{E}(\theta) \triangleq \mathrm{E}_{Y|\Theta}[\hat{\theta}(Y) - \theta)(\hat{\theta}(Y) - \theta)^T]. \tag{5.1}$$

The Fisher information matrix $\mathbf{F}(\theta)$[3] is given by

$$\mathbf{F}_{ij}(\theta) \triangleq \mathrm{E}_{Y|\Theta}\left[\nabla_{\theta_i} \log p(Y|\theta) \nabla^T_{\theta_j} \log p(Y|\theta)\right], \tag{5.2}$$

where $\mathbf{F}_{ij}(\theta)$ is the (i,j)-th element of $\mathbf{F}(\theta)$, $\nabla_v \triangleq \left[\frac{\partial}{\partial v_1}, \ldots, \frac{\partial}{\partial v_q}\right]^T$ with $v \in \mathbb{R}^q$, and $v \triangleq (v_1, \ldots, v_q)^T$. Note that $\mathbf{F}_{ij}$ is a matrix, since the components Θ_k are in general vectors. The Fisher information matrix $\mathbf{F}(\theta)$ can be computed in several ways (Lemma K.6):

$$\begin{aligned} \mathbf{F}_{ij}(\theta) &\triangleq \mathrm{E}_{\Theta|Y}\left[\nabla_{\theta_i} \log p(Y|\theta) \nabla^T_{\theta_j} \log p(Y|\theta)\right] && (5.3) \\ &= -\mathrm{E}_{\Theta|Y}\left[\nabla_{\theta_i} \nabla^T_{\theta_j} \log p(Y|\theta)\right]. && (5.4) \end{aligned}$$

The inverse of the Fisher information matrix is a lower bound on the error matrix $\mathbf{E}(\theta)$ (see, e.g., [199, pp. 66–67], [176, pp. 301–303]).

Theorem 5.1. (Cramér-Rao bound)
Suppose that, for all θ:

a) the error matrix $\mathbf{E}(\theta)$ of the estimator $\hat{\theta}(y)$ exists,

[3]The Fisher information matrix not only plays an important role in statistics, but also in information geometry [7] and machine learning (e.g., [89]) (see Section 5.6).

b) the Fisher information matrix $\mathbf{F}(\theta)$ is non-singular,

c) the support of $p(y|\theta)$ with respect to y does not depend on θ,

d) the pdf $p(y|\theta)$ is differentiable with respect to all coordinates θ_i for all y,

e) the integral $B(\theta) \stackrel{\triangle}{=} \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy$ can be differentiated under the integral sign,

f) the estimator $\hat{\theta}(y)$ is unbiased, i.e., $B(\theta) \stackrel{\triangle}{=} \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy = 0$,

Then

$$\mathbf{E}(\theta) \succeq \mathbf{F}(\theta)^{-1}. \tag{5.5}$$

The inequality (5.5) means that the matrix $\mathbf{D}(\theta) \stackrel{\triangle}{=} \mathbf{E}(\theta) - \mathbf{F}(\theta)^{-1}$ is positive semi-definite. The inequality (5.5) is known as the Cramér-Rao bound, but was in fact first proposed by Fisher in the early days of statistics [63]. Note that the bound merely applies to *unbiased* estimators $\hat{\theta}(y)$. We remind the reader of the fact that the minimum mean square error (MMSE) estimator, i.e., the estimator that minimizes $\mathbf{E}(\theta)$, is *not* necessarily unbiased.[4] This is for example the case if Θ takes values in an interval $[a, b]$ or $[a, \infty)$, as in phase, frequency, timing and noise variance estimation.

Remark 5.1. (Singular information matrix)
The above bound only applies if the Fisher information matrix $\mathbf{F}(\theta)$ is non-singular. What if $\mathbf{F}(\theta)$ is singular? One may then add a diagonal matrix $\mathbf{D}$ to $\mathbf{F}(\theta)$, e.g.,

$$\mathbf{D} \stackrel{\triangle}{=} \varepsilon \mathbf{I}, \tag{5.6}$$

where ε is a "small" positive number, and $\mathbf{I}$ is a unity matrix. If the resulting matrix $\tilde{\mathbf{F}}(\theta) \stackrel{\triangle}{=} \mathbf{F}(\theta) + \mathbf{D}$ is non-singular, it is a lower bound on $\mathbf{E}(\theta)$:

$$\mathbf{E}(\theta) \succeq \tilde{\mathbf{F}}(\theta)^{-1}. \tag{5.7}$$

This applies also to the other Cramér-Rao-type bounds we will encounter in later sections.

[4]In fact, an estimator that mininizes $\mathbf{E}(\theta)$ for all θ may even not exist.

Definition 5.1. (Regularity)
An estimator $\hat{\theta}(y)$ and an estimation problem with conditional $p(y|\theta)$ are called *regular* if the first five assumptions in Theorem 5.1 are met (for all θ). Note that Assumptions 2 through 4 merely concern the estimation problem, more precisely, the pdf $p(y|\theta)$, whereas Assumption 1 and 5 also concern the estimator $\hat{\theta}(y)$. $\square$

In many estimation problems, the observation vector consists of multiple samples drawn from the (same) pdf $p(\cdot|\theta)$; the pdf $p(y|\theta)$ can then be written as:

$$p(y|\theta) \triangleq \prod_{k=1}^{N} p(y_k|\theta), \tag{5.8}$$

where y_k is the k-th observation, and N is the total number of observations. The corresponding error matrix $\mathbf{E}(\theta)$ and Fisher information matrix $\mathbf{F}(\theta)$ depend on N.

For any regular estimator (biased as well as unbiased) and any regular estimation problem of the form (5.8), the Cramér-Rao bounds holds in the limit of an infinite number of observations (i.e., as $N \to \infty$); the Cramér-Rao bound is thus an *asymptotic* bound ("high-SNR bound") for any regular estimator and any regular estimation problem.

Theorem 5.2. (Asymptotic Cramér-Rao bound)
Suppose that:

a) θ takes values in an interval $[a, b]$, with $a, b \in \mathbb{R}$, and $a < b$,

b) the estimator $\hat{\theta}(y)$ is regular, and the estimation problem (with pdf $p(y|\theta)$) is regular,

c) the conditional $p(y|\theta)$ has the form $p(y|\theta) \triangleq \prod_{k=1}^{N} p(y_k|\theta)$ for all $\theta \in [a, b]$,

Then

$$\lim_{N\to\infty} \mathbf{E}(\theta) \succeq \lim_{N\to\infty} \mathbf{F}(\theta)^{-1}, \forall\theta \in (a, b) \tag{5.9}$$

We refer to [75] for a particularly elegant proof of Theorem 5.2. In the limit of an infinite number of observations (and under some additional weak conditions), the estimation error of the ML-estimator, i.e., $\hat{\theta}^{\mathrm{ML}}(y)-$

θ, is a zero-mean Gaussian random vector whose covariance matrix $\mathbf{E}(\theta)$ is given by the inverse of the Fisher information matrix.[5] In other words, the ML-estimator becomes the MMSE estimator as soon as the number of samples is sufficiently large.

Example 5.1. (CRB for the mean of Gaussian random variables) Suppose that we draw N i.i.d. samples $y_1, \ldots, y_N$ from a Gaussian distribution with known variance σ^2, but unknown real-valued mean Θ. We wish to compute the CRB for the estimation of the mean Θ from the N samples $y_1, \ldots, y_N$. The pdf $p(y|\theta)$ equals

$$p(y|\theta) = \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\theta - y_k)^2/2\sigma^2}. \tag{5.10}$$

Since

$$\begin{aligned} \frac{d^2}{d\theta^2} \log p(y|\theta) &= -\frac{1}{2\sigma^2} \sum_{k=1}^{N} \frac{d^2}{d\theta^2} \left[(\theta - y_k)^2\right] && (5.11) \\ &= -\frac{N}{\sigma^2}, && (5.12) \end{aligned}$$

it follows

$$\begin{aligned} \mathbf{F}(\theta) &= -\mathrm{E}_Y \left[\frac{d^2}{d\theta^2} \log p(Y|\theta) \right] && (5.13) \\ &= \frac{N}{\sigma^2}. && (5.14) \end{aligned}$$

The Fisher information matrix (5.14) is a scalar, and it does not depend on Θ. From (5.5) and (5.14), it follows:

$$\mathrm{E}_Y[(\hat{\theta}(Y) - \theta)^2] \geq \sigma^2/N. \tag{5.15}$$

As is well known, the ML-estimator of the mean Θ is given by the empirical mean. Its MSE is equal to σ^2/N; in other words, the empirical mean achieves the CRB (5.15).

Assume now that $\Theta \in [a, b]$ with $a, b \in \mathbb{R}$ and $a < b$. The CRB is again given by (5.15). Note that the CRB grows unboundedly as σ^2 increases,

[5] We refer to [176, pp. 421–424] for a more precise statement of the result, including the proof.

whereas the MSE of any estimator of Θ is bounded, since Θ takes values in a finite interval. In other words, the CRB (5.15) is invalid if $\Theta \in [a, b]$. This is due to the fact that all estimators are necessarily biased, and Condition f of Theorem 5.1 is not fulfilled. On the other hand, the CRB is valid for all $\Theta \in (a, b)$ as $\sigma^2 \to 0$ or $N \to \infty$ (cf. Theorem 5.2). $\square$

5.2.2 Bayesian Cramér-Rao Bounds

We again start by introducing our notation. Let $X = (X_1, X_2, \ldots, X_n)^T$ and $Y = (Y_1, Y_2, \ldots, Y_N)^T$, where X_k and Y_k are real random vectors (the extension to complex random vectors is straightforward); the vectors X_k and Y_k do not necessarily all have the same size. The index k may stand for (discrete) time, i.e., X and Y may be stochastic processes. Suppose $p(x, y)$ is the joint probability density function (pdf) of X and Y. We consider the problem of estimating X from an observation vector $y = (y_1, y_2, \ldots, y_N)^T$. Let the function $\hat{x}(y)$ be an estimator of X based on the observation y. We define the error matrix $\mathbf{E}$ of the estimator $\hat{x}(y)$ as

$$\mathbf{E} \triangleq \mathrm{E}_{XY}[(\hat{x}(Y) - X)(\hat{x}(Y) - X)^T]. \tag{5.16}$$

The Bayesian information matrix $\mathbf{J}$ is given by

$$\mathbf{J}_{ij} \triangleq \mathrm{E}_{XY}\left[\nabla_{x_i} \log p(X, Y) \nabla_{x_j}^T \log p(X, Y)\right]. \tag{5.17}$$

Note that $\mathbf{J}_{ij}$ is a matrix, since the components X_k are in general vectors. The Bayesian information matrix $\mathbf{J}$ can be computed in several ways (Lemmas K.7–K.9)

$$\begin{aligned}
\mathbf{J}_{ij} &\triangleq \mathrm{E}_{XY}\left[\nabla_{x_i} \log p(X, Y) \nabla_{x_j}^T \log p(X, Y)\right] && (5.18)\\
&= -\mathrm{E}_{XY}\left[\nabla_{x_i} \nabla_{x_j}^T \log p(X, Y)\right] && (5.19)\\
&= -\mathrm{E}_{XY}\left[\nabla_{x_i} \nabla_{x_j}^T \log p(Y|X)\right]\\
&\quad -\mathrm{E}_{X}\left[\nabla_{x_i} \nabla_{x_j}^T \log p(X)\right] && (5.20)\\
&= \mathrm{E}_{XY}\left[\nabla_{x_i} \log p(Y|X) \nabla_{x_j}^T \log p(Y|X)\right]\\
&\quad +\mathrm{E}_{X}\left[\nabla_{x_i} \log p(X) \nabla_{x_j}^T \log p(X)\right]. && (5.21)
\end{aligned}$$

The equality (5.19) follows from Lemma K.7, whereas (5.20) is based on the chain rule for probabilities:

$$p(x, y) = p(y|x)p(x). \tag{5.22}$$

The equality (5.21) follows from Lemma K.8 and K.9.

Note that the Bayesian information matrix $\mathbf{J}$ is constant, whereas the Fisher $\mathbf{F}(x)$ information matrix depends on x. In general,

$$\mathbf{J} = \mathrm{E}_X[\mathbf{F}(X)] + \mathrm{E}_X\left[\nabla_x \log p(X) \nabla_x^T \log p(X)\right]. \tag{5.23}$$

If the prior $p(x)$ is uniform, it follows from (5.23):

$$\mathbf{J} = \mathrm{E}_X[\mathbf{F}(X)]. \tag{5.24}$$

If, in addition, $\mathbf{F}(x)$ does not depend on x, i.e., $\mathbf{F}(x) \triangleq \mathbf{F}$, then $\mathbf{J} = \mathbf{F}$.

In '68, Van Trees proved a Cramér-Rao-type bound for random variables [199, pp. 72–73].

Theorem 5.3. (Unconditional Bayesian Cramér-Rao bound)
Suppose that:

a) the error matrix $\mathbf{E}$ of the estimator $\hat{x}(y)$ exists,

b) the Bayesian information matrix $\mathbf{J}$ is non-singular,

c) the support of $p(y|x)$ with respect to y does not depend on x,

d) the pdf $p(x, y)$ is differentiable with respect to all coordinates x_i for all (x, y) belonging to the support of $p(x, y)$,

e) the integral $B(x) \triangleq \int_y [\hat{x}(y) - x] p(y|x) dy$ can be differentiated under the integral sign with respect to all coordinates x_i, $\forall x$,

f) the prior $p(x)$ is zero at the boundary of its support ("weak unbiasedness condition"),

Then

$$\mathbf{E} \succeq \mathbf{J}^{-1}. \tag{5.25}$$

The inequality (5.25) is often referred to as the "Bayesian Cramér-Rao bound" (BCRB), "posterior CRB" or the "Van Trees bound". An estimator $\hat{x}(y)$ and an estimation problem with joint pdf $p(x, y)$ are called *regular* if the first five assumptions in Theorem 5.3 are met. If the joint pdf $p(x, y)$ is Gaussian, the bound (5.25) holds with equality. Note that the BCRB also holds for *biased* estimators, in contrast to the CRB. The weak unbiasedness condition (Assumption 6), however, is not necessarily fulfilled. As for the CRB, this is for example the case when X takes values in an interval $[a, b]$ or $[a, \infty)$, with $a, b \in \mathbb{R}$ and $a < b$. On the other hand, the Bayesian Cramér-Rao bound (5.25) holds at high SNR for *any* regular joint pdf $p(x, y)$, i.e., also for a pdf $p(x, y)$ for which the weak unbiasedness condition is not met. In addition, the MAP estimator achieves the bound (5.25) at high SNR.

The Bayesian Cramér-Rao lower bound (5.25) is in the literature also referred to as *unconditional* Bayesian Cramér-Rao lower bound; in contrast, the *conditional* BCRB bounds the MSE *conditioned* on a particular observation y [24]. The conditional Bayesian information matrix $\mathbf{J}(y)$ is defined as:

$$\mathbf{J}_{ij}(y) \overset{\triangle}{=} \mathrm{E}_{X|Y}\left[\nabla_{x_i} \log p(X|y) \nabla^T_{x_j} \log p(X|y)\right]. \tag{5.26}$$

Since $\nabla_{x_i} \log p(x|y) = \nabla_{x_i} \log p(x, y)$, the matrix (5.26) can also be written as:

$$\mathbf{J}_{ij}(y) = \mathrm{E}_{X|Y}\left[\nabla_{x_i} \log p(X, y) \nabla^T_{x_j} \log p(X, y)\right]. \tag{5.27}$$

From Lemma K.6 follows an alternative expression for the matrix (5.26):

$$\begin{aligned} \mathbf{J}_{ij}(y) &= -\mathrm{E}_{X|Y}\left[\nabla_{x_i} \nabla^T_{x_j} \log p(X|y)\right] & (5.28)\\ &= -\mathrm{E}_{X|Y}\left[\nabla_{x_i} \nabla^T_{x_j} \log p(X, y)\right]. & (5.29) \end{aligned}$$

The inverse conditional Bayesian information matrix is a lower bound on the error matrix $\mathbf{E}(y)$ defined as:

$$\mathbf{E}(y) \overset{\triangle}{=} \mathrm{E}_{X|Y}[(\hat{x}(y) - X)(\hat{x}(y) - X)^T]. \tag{5.30}$$

Theorem 5.4. (Conditional Bayesian Cramér-Rao bound)
Suppose that:

a) the error matrix $\mathbf{E}(y)$ of the estimator $\hat{x}(y)$ exists,

b) the conditional Bayesian information matrix $\mathbf{J}(y)$ is non-singular,

c) the pdf $p(x|y)$ is differentiable with respect to all coordinates x_i for all (x, y) belonging to the support of $p(x|y)$,

d) the prior $p(x)$ is zero at the boundary of its support ("weak unbiasedness condition"),

Then

$$\mathbf{E}(y) \succeq \mathbf{J}^{-1}(y). \tag{5.31}$$

If $p(x|y)$ is Gaussian, the bound (5.31) holds with equality.

From (5.31), an alternative lower bound on $\mathbf{E}$ can be derived.[6]

Corollary 5.1. (Alternative unconditional Bayesian Cramér-Rao bound) Suppose that:

a) the error matrix $\mathbf{E}(y)$ of the estimator $\hat{x}(y)$ exists for all y,

b) the Bayesian information matrix $\mathbf{J}(y)$ is non-singular for all y,

c) the pdf $p(x, y)$ is differentiable with respect to all coordinates x_i for all (x, y) belonging to the support of $p(x, y)$,

d) the prior $p(x)$ is zero at the boundary of its support ("weak unbiasedness condition"),

Then

$$\mathbf{E} \succeq \mathrm{E}_Y \left[\mathbf{J}^{-1}(Y)\right]. \tag{5.32}$$

The bound (5.32) is tighter than the bound (5.25).

Lemma 5.1. If $\mathbf{J}$ and $\mathbf{J}(y)$ (for all y) are non-singular, then

$$\mathbf{J}^{-1} \preceq \mathrm{E}_Y \left[\mathbf{J}^{-1}(Y)\right]. \tag{5.33}$$

$\square$

[6]To our knowledge, this bound is novel.

If the pdf $p(x, y)$ is Gaussian, the bounds (5.25) and (5.32) coincide, since the matrix $\mathbf{J}(y)$ is then independent of Y.

From the CRB (5.5), one can also derive an unconditional BCRB ("Fisher-Bayesian Cramér-Rao bound") [206].

Theorem 5.5. (Fisher-Bayesian Cramér-Rao bound)
Suppose that, for all x,

a) the estimator $\hat{\theta}(y)$ is regular, and the estimation problem (with pdf $p(y|x)$) is regular,

b) the estimator $\hat{x}(y)$ is unbiased, i.e., $B(x) \stackrel{\triangle}{=} \int_y [\hat{x}(y) - x] p(y|x) dy = 0$,

Then

$$\mathbf{E} \succeq \mathrm{E}_X \left[\mathbf{F}^{-1}(X) \right]. \tag{5.34}$$

The bound (5.34) is tighter than the bound (5.25).

Lemma 5.2. If $\mathbf{J}$ and $\mathbf{F}(x)$ (for all x) are non-singular, then

$$\mathbf{J}^{-1} \preceq \mathrm{E}_X \left[\mathbf{F}^{-1}(X) \right]. \tag{5.35}$$

□

The Fisher BCRB (5.34) only holds for unbiased estimators, whereas the standard and alternative unconditional BCRBs (5.25) and (5.32) also hold for biased estimators (as long as $p(x)$ is zero at the boundary of its support). If the prior $p(x)$ is non-trivial, the MMSE-estimator is most often biased and the Fisher-Bayesian Cramér-Rao bound (5.34) is not applicable (at finite SNR). If the pdf $p(x, y)$ is Gaussian, and the prior $p(x)$ is uniform, i.e.,

$$p(x, y) \propto p(y|x), \tag{5.36}$$

the bounds (5.25), (5.32), and (5.34) coincide, since the matrices $\mathbf{F}(x)$ and $\mathbf{J}(y)$ are then independent of X and Y respectively.

Example 5.2. (BCRB for mean of Gaussian random variables)
We consider again the estimation problem of Example 5.1. Now, we

suppose that the (unknown) real-valued mean is a random variable (with a non-trivial prior). We denote the mean by X, and its prior by $p(x)$. We wish to compute the BCRBs (5.25), (5.32) and (5.34) for estimating X from $y = y_1, \ldots, y_N$. The joint pdf $p(x, y)$ equals:

$$p(x,y) = p(x) \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-y_k)^2/2\sigma^2}. \tag{5.37}$$

Since

$$\begin{aligned} \frac{d^2}{dx^2} \log p(x,y) &= \frac{d^2}{dx^2} \log p(y|x) + \frac{d^2}{dx^2} \log p(x) && (5.38) \\ &= -\frac{N}{\sigma^2} + \frac{d^2}{dx^2} \log p(x), && (5.39) \end{aligned}$$

it follows

$$\begin{aligned} \mathbf{J} &= -\mathrm{E}_{XY}\left[\frac{d^2}{dx^2} \log p(X,Y)\right] && (5.40) \\ &= \frac{N}{\sigma^2} - \mathrm{E}_X\left[\frac{d^2}{dx^2} \log p(X)\right], && (5.41) \end{aligned}$$

and

$$\begin{aligned} \mathbf{J}(y) &= -\mathrm{E}_{X|Y}\left[\frac{d^2}{dx^2} \log p(X,y)\right] && (5.42) \\ &= \frac{N}{\sigma^2} - \mathrm{E}_{X|Y}\left[\frac{d^2}{dx^2} \log p(X)\right]. && (5.43) \end{aligned}$$

Note that $\mathbf{J}$ and $\mathbf{J}(y)$ are scalar. From (5.25) and (5.41) follows the standard unconditional BCRB:

$$\mathrm{E}_{XY}[(\hat{x}(X) - X)^2] \geq \left(\frac{N}{\sigma^2} - \mathrm{E}_X\left[\frac{d^2}{dx^2} \log p(X)\right]\right)^{-1}. \tag{5.44}$$

From (5.32) and (5.43) follows the alternative unconditional BCRB:

$$\mathrm{E}_{XY}[(\hat{x}(Y) - X)^2] \geq \mathrm{E}_Y\left[\left(\frac{N}{\sigma^2} - \mathrm{E}_{X|Y}\left[\frac{d^2}{dx^2} \log p(X)\right]\right)^{-1}\right]. \tag{5.45}$$

From (5.15) and (5.34) follows the Fisher BCRB:

$$\mathrm{E}_{XY}[(\hat{x}(Y) - X)^2] \geq \left(\frac{N}{\sigma^2}\right)^{-1}. \tag{5.46}$$

The Fisher BCRB (5.46) is only valid for unbiased estimators, whereas the MMSE estimator is usually biased if the prior $p(x)$ is non-uniform. If the prior $p(x)$ is uniform, the bounds (5.44) and (5.45) reduce to the Fisher BCRB (5.46). Note that

$$-\mathrm{E}_X\left[\frac{d^2}{dx^2}\log p(X)\right] = \mathrm{E}_X\left[\left(\frac{d}{dx}\log p(X)\right)^2\right] \geq 0, \tag{5.47}$$

and therefore (cf. Lemma K.4),

$$\left(\frac{N}{\sigma^2} - \mathrm{E}_X\left[\frac{d^2}{dx^2}\log p(X)\right]\right)^{-1} \leq \left(\frac{N}{\sigma^2}\right)^{-1}. \tag{5.48}$$

Similarly,

$$\left(\frac{N}{\sigma^2} - \mathrm{E}_{X|Y}\left[\frac{d^2}{dx^2}\log p(X)\right]\right)^{-1} \leq \left(\frac{N}{\sigma^2}\right)^{-1}. \tag{5.49}$$

In words: if the prior $p(x)$ is non-uniform, the BCRBs (5.44) and (5.45) are lower than the bound (5.46), which is valid for a uniform prior $p(x)$. This makes sense: a non-uniform prior $p(x)$ contains additional information about X; the more informative the prior, the smaller the estimation error will be. Not surprisingly, the BCRB (5.44) and (5.45) reduce to (5.46) as $N \to \infty$ or $\sigma^2 \to 0$: as the observations become more informative, the prior knowledge loses its importance.

Assume now that $X \in [a, b]$, with $a, b \in \mathbb{R}$ and $a < b$. The standard BCRB, alternative BCRB and Fisher BCRB are again given by (5.44), (5.45) and (5.46) respectively. Suppose that $p(x)$ is uniform, i.e.,

$$p(x) = \frac{1}{b-a}, \quad \forall x \in [a, b]. \tag{5.50}$$

Both BCRBs (5.44) and (5.45) reduce then to the Fisher BCRB (5.46). Note that the BCRB (5.46) (as the CRB (5.15)) grows unboundedly as σ^2 increases, whereas the MSE of any estimator $\hat{x}(y)$ of $X \in [a, b]$ is obviously bounded. In other words, the BCRBs (5.44) and (5.45) are invalid for the prior (5.50). This is due to the fact that

$$p(a) \neq 0 \neq p(b). \tag{5.51}$$

Nevertheless, the BCRBs (5.44) and (5.45) become valid as $N \to \infty$ or $\sigma^2 \to 0$ ("high SNR bound").

$\square$

Bayesian Cramér-Rao Bounds and Marginalization

In practice, one is often interested in bounding the MSE for a particular variable X_k, i.e., for a particular *component* of the vector $X = (X_1, \ldots, X_n)^T$. For example, one may wish to compute a BCRB for the MSE:

$$\begin{aligned} &\mathrm{E}_{X_k Y}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] \\ &\quad = \mathrm{E}_{XY}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] \qquad (5.52) \\ &\quad \stackrel{\triangle}{=} \mathbf{E}_{kk}, \qquad (5.53) \end{aligned}$$

which is the k-th diagonal element of the error matrix $\mathbf{E}$. Of practical relevance is also the (weighted) average of the MSE over all components of X:

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk} \stackrel{\triangle}{=} \sum_{k=1}^{n} w_k \mathrm{E}_{X_k Y}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T], \qquad (5.54)$$

where n is the dimension of X, and w_k is a positive real number (typically, $w_k = 1/n$).

There are several ways to obtain a BCRB for (5.52) and (5.54). One may derive a BCRB from the information matrix of the *joint* pdf $p(x, y)$ (or $p(x|y)$). For example, from the standard unconditional BCRB (5.25), it follows:

$$\mathbf{E}_{kk} \succeq [\mathbf{J}^{-1}]_{kk}, \qquad (5.55)$$

and

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk} \succeq \sum_{k=1}^{n} w_k [\mathbf{J}^{-1}]_{kk}, \qquad (5.56)$$

where the unconditional Bayesian information matrix $\mathbf{J}$ is computed from the joint pdf $p(x, y)$. Note that in the RHS of (5.55) and (5.56), only the *diagonal* elements of $\mathbf{J}^{-1}$ appear, the off-diagonal elements of $\mathbf{J}^{-1}$ are not required. Along similar lines, an alternative BCRB for $\mathbf{E}_{kk}$ (5.52) and the average (5.54) can be derived from (5.32):

$$\mathbf{E}_{kk} \succeq \mathrm{E}_Y\big[[\mathbf{J}^{-1}(Y)]_{kk}\big], \qquad (5.57)$$

and

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk} \succeq \sum_{k=1}^{n} w_k \mathrm{E}_Y\big[[\mathbf{J}^{-1}(Y)]_{kk}\big], \qquad (5.58)$$

where the conditional Bayesian information matrix $\mathbf{J}(y)$ is computed from the joint pdf $p(x|y)$. Similarly, (B)CRBs for $\mathbf{E}_{kk}(x)$ and $\mathbf{E}_{kk}(y)$ can be derived from (5.5) and (5.31) respectively.

Alternatively, instead of deriving the BCRB from the information matrix of $p(x, y)$ (or $p(x|y)$) (cf. (5.55) to (5.58)), one may first *marginalize* over some variables X_ℓ ($\ell \neq k$), and compute the BCRB from the information matrix of the resulting *marginal* of $p(x, y)$ (or $p(x|y)$). Let us have a look at a simple example.

Example 5.3. (Marginalization and BCRB)
Let $X \triangleq (X_1, X_2)^T$, and hence $p(x, y) \triangleq p(x_1, x_2, y)$. Suppose that we wish to obtain a standard unconditional BCRB for X_1. This can be done in two ways. One may compute the unconditional Bayesian information matrix of $p(x_1, y)$; the inverse of that matrix is a standard unconditional BCRB for X_1:

$$\begin{aligned}&\mathrm{E}_{X_1Y}[(\hat{x}_1(Y) - X_1)(\hat{x}_1(Y) - X_1)^T] \\ &\qquad \succeq \mathrm{E}_{X_1Y}\left[\nabla_{x_1}\nabla_{x_1}^T \log p(X_1, Y)\right]^{-1},\end{aligned} \tag{5.59}$$

where:

$$p(x_1, y) \triangleq \int_{x_2} p(x_1, x_2, y)dx_2. \tag{5.60}$$

Alternatively, one may derive a standard unconditional BCRB from the unconditional Bayesian information matrix of $p(x_1, x_2, y)$, which is a 2×2 block matrix. More precisely, the first diagonal element of the inverse of that matrix is a standard unconditional BCRB for X_1:

$$\mathrm{E}_{X_1Y}[(\hat{x}_1(Y) - X_1)(\hat{x}_1(Y) - X_1)^T] \succeq \left[\mathbf{J}^{-1}\right]_{11}, \tag{5.61}$$

where

$$\mathbf{J} \triangleq \begin{bmatrix} -\mathrm{E}_{X_1X_2Y}\big[\nabla_{x_1}\nabla_{x_1}^T \log p(X_1, X_2, Y)\big] & -\mathrm{E}_{X_1X_2Y}\big[\nabla_{x_1}\nabla_{x_2}^T \log p(X_1, X_2, Y)\big] \\ -\mathrm{E}_{X_1X_2Y}\big[\nabla_{x_2}\nabla_{x_1}^T \log p(X_1, X_2, Y)\big] & -\mathrm{E}_{X_1X_2Y}\big[\nabla_{x_2}\nabla_{x_2}^T \log p(X_1, X_2, Y)\big] \end{bmatrix}. \tag{5.62}$$

□

Bayesian Cramér-Rao bounds for a variable X_k can thus be derived in various ways, since one has the freedom to marginalize some variables X_ℓ ($\ell \neq k$) before computing the required information matrix. Which approach leads to the tighest bounds? For example, is the bound (5.59) tighter than (5.61)? An answer to those questions is given in [24] (see also [170]): the tightest Bayesian Cramér-Rao bound for a variable X_k is obtained by first marginalizing over all variables X_ℓ ($\ell \neq k$), and by then computing the inverse information matrix of the resulting marginal $p(x_k, y)$ (or $p(x_k|y)$). For instance, the bound (5.59) is tighter than (5.61). It is typically easier, however, to derive Cramér-Rao-type bounds from the joint pdf (as in (5.61)) than from a marginal (as in (5.59)).[7]

Remark 5.2. (A common misunderstanding)
Some researchers (e.g., [148]) believe that

- Cramér-Rao-type bounds derived from joint pdfs only apply to estimators that estimate all unknown variables *jointly*,
- Cramér-Rao-type bounds derived from marginal pdfs only apply to estimators that treat some of the variables as *nuisance parameters*; the latter are not estimated explicitly, instead, they are eliminated by marginalization.

For example, the bound (5.61) for the MSE of X_1 would only hold for estimators that jointly estimate X_1 and X_2, whereas the bound (5.59) would only hold for estimators that *marginalize* over X_2 (instead of estimating X_2). According to the same researchers (e.g., [148]), the fact that the bound (5.59) is tighter than (5.61) would imply that one should estimate X_1 and X_2 *jointly* in order to obtain the smallest MSE for X_1, since treating X_2 as a nuisance parameter leads to a higher MSE for X_1.

This is a misconception: both BCRBs (5.59) and (5.61) hold for the MMSE estimator of X_1 (assuming that the necessary conditions are fulfilled), and hence for *any* estimator of X_1.

5.2.3 Hybrid Cramér-Rao Bounds

We now consider systems that contain random variables X as well as parameters Θ; the joint pdf of such a system is given by $p(x, y|\theta)$. We define the error matrix $\mathbf{E}^{(X\Theta)}(\theta)$ as

$$\mathbf{E}^{(X\Theta)}(\theta) \triangleq \begin{bmatrix} \mathbf{E}_{11}^{(X\Theta)}(\theta) & \mathbf{E}_{12}^{(X\Theta)}(\theta) \\ \mathbf{E}_{21}^{(X\Theta)}(\theta) & \mathbf{E}_{22}^{(X\Theta)}(\theta) \end{bmatrix}, \tag{5.63}$$

[7]See Example 5.9 and 5.10.

where:

$$\begin{aligned}
\mathbf{E}_{11}^{(X\Theta)}(\theta) &= \mathrm{E}_{XY|\Theta}\Big[(\hat{\theta}(Y)-\theta)(\hat{\theta}(Y)-\theta)^T\Big] && (5.64)\\
&= \mathrm{E}_{Y|\Theta}\Big[(\hat{\theta}(Y)-\theta)(\hat{\theta}(Y)-\theta)^T\Big] && (5.65)\\
\mathbf{E}_{12}^{(X\Theta)}(\theta) &= \mathrm{E}_{XY|\Theta}\Big[(\hat{\theta}(Y)-\theta)(\hat{x}(Y)-X)^T\Big] && (5.66)\\
\mathbf{E}_{21}^{(X\Theta)}(\theta) &= [\mathbf{E}_{12}^{(X\Theta)}(\theta)]^T && (5.67)\\
\mathbf{E}_{22}^{(X\Theta)}(\theta) &= \mathrm{E}_{XY|\Theta}\big[(\hat{x}(Y)-X)(\hat{x}(Y)-X)^T\big]. && (5.68)
\end{aligned}$$

The "hybrid" unconditional information matrix $\mathbf{H}(\theta)$ is defined as:

$$\mathbf{H}(\theta) \triangleq \begin{bmatrix} \mathbf{H}_{11}(\theta) & \mathbf{H}_{12}(\theta) \\ \mathbf{H}_{21}(\theta) & \mathbf{H}_{22}(\theta) \end{bmatrix}, \qquad (5.69)$$

where:

$$\begin{aligned}
\mathbf{H}_{11}(\theta) &= \mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta)\nabla_\theta^T \log p(X,Y|\theta)\right] && (5.70)\\
\mathbf{H}_{12}(\theta) &= \mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta)\nabla_x^T \log p(X,Y|\theta)\right] && (5.71)\\
\mathbf{H}_{12}(\theta) &= [\mathbf{H}_{21}(\theta)]^T && (5.72)\\
\mathbf{H}_{22}(\theta) &= \mathrm{E}_{XY|\Theta}\left[\nabla_x \log p(X,Y|\theta)\nabla_x^T \log p(X,Y|\theta)\right]. && (5.73)
\end{aligned}$$

The elements of the hybrid unconditional information matrix $\mathbf{H}(\theta)$ can also be written as:

$$\begin{aligned}
\mathbf{H}_{11}(\theta) &= -\mathrm{E}_{XY|\Theta}\left[\nabla_\theta\nabla_\theta^T \log p(X,Y|\theta)\right] && (5.74)\\
\mathbf{H}_{12}(\theta) &= -\mathrm{E}_{XY|\Theta}\left[\nabla_\theta\nabla_x^T \log p(X,Y|\theta)\right] && (5.75)\\
\mathbf{H}_{22}(\theta) &= -\mathrm{E}_{XY|\Theta}\left[\nabla_x\nabla_x^T \log p(X,Y|\theta)\right]. && (5.76)
\end{aligned}$$

The inverse of the hybrid unconditional information matrix $\mathbf{H}(\theta)$ is a lower bound on the error matrix $\mathbf{E}^{(X\Theta)}(\theta)$.

Theorem 5.6. (Hybrid unconditional Bayesian Cramér-Rao bound [172]; see also [170])
Suppose that:

a) the error matrix $\mathbf{E}^{(X\Theta)}(\theta)$ of the estimator $(\hat{\theta}(y), \hat{x}(y))$ exists, $\forall\theta$,

b) the hybrid unconditional information matrix $\mathbf{H}(\theta)$ is non-singular, $\forall\theta$,

c) the support of $p(y|x,\theta)$ with respect to y does not depend on x and θ, $\forall x$ and θ,

d) the pdf $p(x,y|\theta)$ is differentiable with respect to all coordinates x_i and θ_i, for all (x,y,θ) belonging to the support of $p(x,y|\theta)$,

e) the integral $B^{(X)}(x) \triangleq \int_y [\hat{x}(y) - x] p(y|x,\theta) dy$ can be differentiated under the integral sign with respect to all coordinates x_i, $\forall x$ and θ,

f) the prior $p(x)$ is zero at the boundary of its support ("weak unbiasedness condition"),

g) the estimator $\hat{\theta}(y)$ is unbiased, i.e., $B^{(\Theta)}(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy = 0, \forall \theta$,

h) the integral $B^{(\Theta)}(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy$ can be differentiated under the integral sign with respect to all coordinates θ_i, $\forall \theta$,

i) the integral $B(\theta) \triangleq \int_{x,y} [\hat{x}(y) - x] p(x,y|\theta) dx dy$ ("average bias") is independent of θ, and can be differentiated under the integral sign with respect to all coordinates θ_i, $\forall \theta$,

Then

$$\mathbf{E}^{(X\Theta)}(\theta) \succeq \mathbf{H}(\theta)^{-1}. \tag{5.77}$$

From (5.77), a Cramér-Rao-type bound for Θ can be obtained as:

$$\mathrm{E}_{Y|\Theta}[(\hat{\theta}(Y) - \theta)(\hat{\theta}(Y) - \theta)^T] \triangleq \mathbf{E}_{11}^{(X\Theta)} \succeq \left[\mathbf{H}^{-1}(\theta)\right]_{11}, \tag{5.78}$$

From (5.77), one also obtains an unconditional Bayesian Cramér-Rao bound for the MSE of X:

$$\mathrm{E}_{XY|\Theta}[(\hat{x}(Y) - X)(\hat{x}(Y) - X)^T] \triangleq \mathbf{E}_{22}^{(X\Theta)} \succeq \left[\mathbf{H}^{-1}(\theta)\right]_{22}. \tag{5.79}$$

Hybrid CRB and Marginalization

Again, one has the freedom to marginalize over certain variables X_ℓ before computing the information matrix. For example, an alternative Cramér-Rao-type bound for Θ is given by:

$$\begin{aligned}\mathrm{E}_{Y|\Theta}[(\hat{\theta}(Y) - \theta)(\hat{\theta}(y) - \theta)^T] &\triangleq \mathbf{E}_{11}^{(X\Theta)} \\ &\succeq \mathrm{E}_{Y|\Theta}\left[\nabla_\theta \nabla_\theta^T \log p(Y|\theta)\right],\end{aligned} \tag{5.80}$$

where

$$p(y|\theta) \triangleq \int_x p(x,y|\theta)dx. \tag{5.81}$$

The CRB (5.80) for Θ is derived by first marginalizing over X. The bound (5.80) is tighter than (5.78) [170].

Remark 5.3. (Marginalization over parameters)
We consider here an alternative to the bound (5.59). One may wish to marginalize over Θ before computing the information matrix. We define $p(x,\theta,y) \triangleq \gamma^{-1}p(x,y|\theta)$, where

$$\gamma \triangleq \int_{x,\theta} p(x,y|\theta)dxd\theta dy, \tag{5.82}$$

assuming that this integral converges. Now, Θ can be treated as a random variable, and one can compute the bound (5.59) with $X_2 \triangleq \Theta$:

$$\begin{aligned} &\mathrm{E}_{XY}[(\hat{x}(Y)-X)(\hat{x}(Y)-X)^T] \\ &\qquad \succeq \mathrm{E}_{XY}\left[\nabla_x\nabla_x^T \log p(X,Y)\right]^{-1}, \end{aligned} \tag{5.83}$$

where:

$$p(x,y) \triangleq \int_\theta p(x,\theta,y)d\theta. \tag{5.84}$$

Note that the error matrix in the LHS of (5.79) depends on Θ, whereas the the error matrix in the LHS of (5.83) does *not* depend on Θ. Both definitions of the error matrix make sense; it depends on the estimation problem at hand which definition is the most appropriate.

5.2.4 Summary

There are two general strategies to obtain Cramér-Rao type bounds for a variable X_k:

a) One computes the inverse information matrix of the *joint* pdf. The bound is then given by a diagonal element of that matrix.

b) One marginalizes over certain (perhaps all) random variables $X_\ell \neq X_k$ before computing the information matrix. The bound is given by a diagonal element of the inverse information matrix of the resulting *marginal*.

Note that the information matrix of the joint pdf is typically large, but *sparse*; this sparseness can be exploited while computing the diagonal elements. Information matrices of marginals are smaller, but they are usually *dense*. Cramér-Rao-Type bounds obtained from marginals are *tighter* than the corresponding bounds derived from the joint pdf.

In the following, we propose practical algorithms for each of the two stategies. In Section 5.3, we describe a summary-propagation algorithm to compute Cramér-Rao-type bounds from the information matrix of the joint pdf (Strategy 1). In Section 5.4, we develop algorithms for computing Cramér-Rao-type bounds from information matrices of marginals (Strategy 2).

5.3 Cramér-Rao-Type Bounds From Joint Densities

5.3.1 Standard Unconditional BCRBs

Suppose we wish to compute the standard unconditional Bayesian Cramér-Rao bound for the MSE of a variable X_k (cf. (5.55)):

$$\mathbf{E}_{kk} \succeq [\mathbf{J}^{-1}]_{kk}, \tag{5.85}$$

or, the standard unconditional Bayesian Cramér-Rao bound for the MSE averaged over all variables X_k (cf. (5.56)):

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk} \succeq \sum_{k=1}^{n} w_k [\mathbf{J}^{-1}]_{kk}. \tag{5.86}$$

In the RHS of (5.85) and (5.86), the inverse of the (potentially huge!) matrix $\mathbf{J}$ occurs. However:

a) Only the diagonal elements of this inverse are required.

b) The joint probability density $p(x, y)$ has in most practical systems a “nice” structure, i.e., $p(x, y)$ has typically a non-trivial factorization. As a consequence, $\mathbf{J}$ is often sparse.

c) This sparseness can effectively be exploited by applying the matrix inversion lemma (Lemma K.2).

As a consequence, the elements $[\mathbf{J}^{-1}]_{kk}$ can be determined by local computations that involve the inversion of matrices that are much smaller than $\mathbf{J}$. Those computations can be viewed as message passing ("summary propagation") on a (cycle-free) factor graph of $p(x, y)$. The summary propagation procedure is similar to the sum(mary)-product algorithm; messages (which are in this case matrices) propagate on the factor graph of $p(x, y)$. They are updated at the nodes according to some rules. The expression $[\mathbf{J}^{-1}]_{kk}$ is obtained from the messages along the edge X_k. In the following, we first investigate a small working example, from which we then extract the general summary-propagation procedure.

Suppose the pdf $p(x, y)$ is given by

$$\begin{aligned} p(x,y) &= \bigg(\Big(\big(f_1(x_1,y_1)f_2(x_1,x_2,x_3)\big)f_3(x_3,y_3)\Big)f_4(x_4,y_4) \\ &\qquad \cdot f_5(x_5,y_5)f_6(x_3,x_4,x_5,x_6)\bigg)\big(f_7(x_7,y_7)f_8(x_6,x_7,x_8)\big), \quad (5.87) \end{aligned}$$

as shown in Fig. 5.1. The brackets in (5.87) correspond to the boxes in Fig. 5.1. The unconditional Bayesian information matrix of (5.87) equals:

$$\mathbf{J} = \begin{bmatrix} \mathbf{f}_1^{11}+\mathbf{f}_2^{11} & \mathbf{f}_2^{12} & \mathbf{f}_2^{13} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{f}_2^{21} & \mathbf{f}_2^{22} & \mathbf{f}_2^{23} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{f}_2^{31} & \mathbf{f}_2^{32} & \mathbf{f}_2^{33}+\mathbf{f}_3^{33}+\mathbf{f}_6^{33} & \mathbf{f}_6^{34} & \mathbf{f}_6^{35} & \mathbf{f}_6^{36} & 0 & 0 \\ 0 & 0 & \mathbf{f}_6^{43} & \mathbf{f}_4^{44}+\mathbf{f}_6^{44} & \mathbf{f}_6^{45} & \mathbf{f}_6^{46} & 0 & 0 \\ 0 & 0 & \mathbf{f}_6^{53} & \mathbf{f}_6^{54} & \mathbf{f}_5^{55}+\mathbf{f}_6^{55} & \mathbf{f}_6^{56} & 0 & 0 \\ 0 & 0 & \mathbf{f}_6^{63} & \mathbf{f}_6^{64} & \mathbf{f}_6^{65} & \mathbf{f}_6^{66}+\mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{f}_8^{76} & \mathbf{f}_7^{77}+\mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix}, \quad (5.88)$$

where we used the notation:

$$\begin{aligned} \mathbf{f}_k^{ij} &\triangleq -\mathrm{E}_{XY}[\nabla_{x_i}\nabla_{x_j}^T \log f_k(X,Y)] & (5.89) \\ &= -\int_{x,y} p(x,y)\nabla_{x_i}\nabla_{x_j}^T \log f_k(x,y)dxdy. & (5.90) \end{aligned}$$

Suppose we wish to compute the matrix $[\mathbf{J}^{-1}]_{66}$, which is a lower bound on the mean square estimation error of X_6 (see (5.55)):

$$\mathrm{E}_{XY}[(\hat{X}_6(Y) - X_6)(\hat{X}_6(Y) - X_6)^T] \triangleq \mathbf{E}_{66} \succeq [\mathbf{J}^{-1}]_{66}. \tag{5.91}$$

We define $\mathbf{J}^{(k\ell)} \triangleq ([\mathbf{J}^{-1}]_{k:\ell})^{-1}$, where $\mathbf{A}_{k:\ell}$ is a submatrix of $\mathbf{A}$ that consists of the rows and columns $k, k+1, \ldots, \ell$ of $\mathbf{A}$. By applying the matrix inversion lemma to the matrix (5.88) (cf. Lemma K.2, with $\mathbf{A} \triangleq \mathbf{J}$ and diagonal submatrices $\mathbf{A}_{11} \triangleq \mathbf{J}_{1:2}$ and $\mathbf{A}_{22} \triangleq \mathbf{J}_{3:8}$), we have:

$$\mathbf{J}^{(38)} \triangleq ([\mathbf{J}^{-1}]_{3:8})^{-1} \tag{5.92}$$

$$= \begin{bmatrix} \mathbf{f}_2^{33} + \mathbf{f}_3^{33} + \mathbf{f}_6^{33} & \mathbf{f}_6^{34} & \mathbf{f}_6^{35} & \mathbf{f}_6^{36} & 0 & 0 \\ \mathbf{f}_6^{43} & \mathbf{f}_4^{44} + \mathbf{f}_6^{44} & \mathbf{f}_6^{45} & \mathbf{f}_6^{46} & 0 & 0 \\ \mathbf{f}_6^{53} & \mathbf{f}_6^{54} & \mathbf{f}_5^{55} + \mathbf{f}_6^{55} & \mathbf{f}_6^{56} & 0 & 0 \\ \mathbf{f}_6^{63} & \mathbf{f}_6^{64} & \mathbf{f}_6^{65} & \mathbf{f}_6^{66} + \mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ 0 & 0 & 0 & \mathbf{f}_8^{76} & \mathbf{f}_7^{77} + \mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ 0 & 0 & 0 & \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix} -$$

$$\begin{bmatrix} \mathbf{f}_2^{31} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{f}_2^{32} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{f}_1^{11} + \mathbf{f}_2^{11} & \mathbf{f}_2^{12} \\ \mathbf{f}_2^{21} & \mathbf{f}_2^{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f}_2^{31} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{f}_2^{32} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.93}$$

$$= \begin{bmatrix} \boldsymbol{\mu}_{f_2 \to X_3} + \mathbf{f}_3^{33} + \mathbf{f}_6^{33} & \mathbf{f}_6^{34} & \mathbf{f}_6^{35} & \mathbf{f}_6^{36} & 0 & 0 \\ \mathbf{f}_6^{43} & \mathbf{f}_4^{44} + \mathbf{f}_6^{44} & \mathbf{f}_6^{45} & \mathbf{f}_6^{46} & 0 & 0 \\ \mathbf{f}_6^{53} & \mathbf{f}_6^{54} & \mathbf{f}_5^{55} + \mathbf{f}_6^{55} & \mathbf{f}_6^{56} & 0 & 0 \\ \mathbf{f}_6^{63} & \mathbf{f}_6^{64} & \mathbf{f}_6^{65} & \mathbf{f}_6^{66} + \mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ 0 & 0 & 0 & \mathbf{f}_8^{76} & \mathbf{f}_7^{77} + \mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ 0 & 0 & 0 & \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix}, \tag{5.94}$$

where

$$\boldsymbol{\mu}_{f_2 \to X_3} \triangleq \left(\left(\begin{bmatrix} \mathbf{f}_1^{11} + \mathbf{f}_2^{11} & \mathbf{f}_2^{12} & \mathbf{f}_2^{13} \\ \mathbf{f}_2^{21} & \mathbf{f}_2^{22} & \mathbf{f}_2^{23} \\ \mathbf{f}_2^{31} & \mathbf{f}_2^{32} & \mathbf{f}_2^{33} \end{bmatrix}^{-1} \right)_{33} \right)^{-1} \tag{5.95}$$

$$= \mathbf{f}_2^{33} - \begin{bmatrix} \mathbf{f}_2^{31} & \mathbf{f}_2^{32} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^{11} + \mathbf{f}_2^{11} & \mathbf{f}_2^{12} \\ \mathbf{f}_2^{21} & \mathbf{f}_2^{22} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{f}_2^{31} & \mathbf{f}_2^{32} \end{bmatrix}^T. \tag{5.96}$$

Also in (5.96), we used the matrix inversion lemma. The matrix $\boldsymbol{\mu}_{f_2 \to X_3}$ summarizes the smallest dashed box on the left in Fig. 5.1. We now

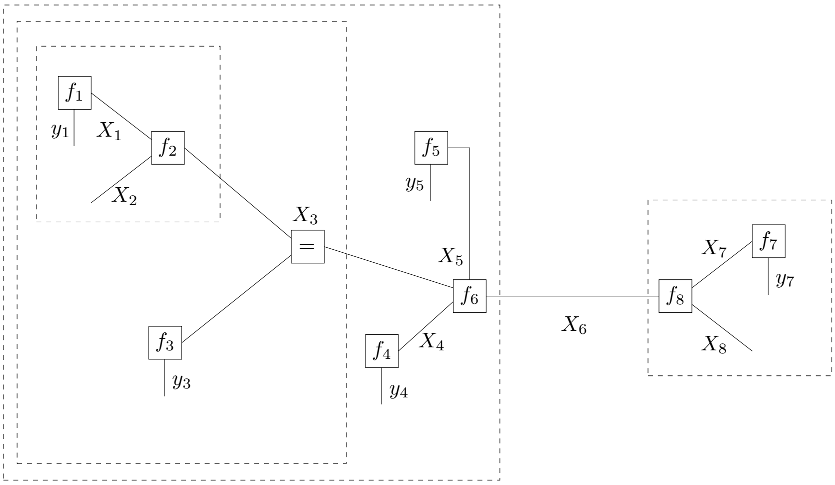


Figure 5.1: Factor graph of (5.87).

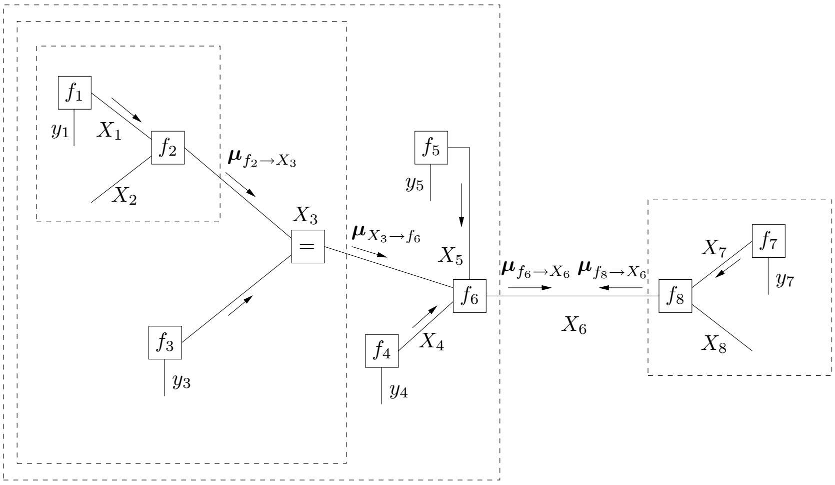

f_1
y_1
X_1
f_2
X_2
$\boldsymbol{\mu}_{f_2 \to X_3}$
X_3
$=$
$\boldsymbol{\mu}_{X_3 \to f_6}$
f_3
y_3
f_5
y_5
X_5
f_6
$\boldsymbol{\mu}_{f_6 \to X_6}$
$\boldsymbol{\mu}_{f_8 \to X_6}$
f_4
X_4
y_4
X_6
X_7
f_7
y_7
f_8
X_8

Figure 5.2: Summary propagation for computing BCRB.

define

$$\boldsymbol{\mu}_{X_3 \to f_6} \triangleq \boldsymbol{\mu}_{f_2 \to X_3} + \mathbf{f}_3^{33}. \tag{5.97}$$

As a consequence

$$\mathbf{J}^{(38)} = \begin{bmatrix} \boldsymbol{\mu}_{X_3\to f_6} + \mathbf{f}_6^{33} & \mathbf{f}_6^{34} & \mathbf{f}_6^{35} & \mathbf{f}_6^{36} & 0 & 0 \\ \mathbf{f}_6^{43} & \mathbf{f}_4^{44} + \mathbf{f}_6^{44} & \mathbf{f}_6^{45} & \mathbf{f}_6^{46} & 0 & 0 \\ \mathbf{f}_6^{53} & \mathbf{f}_6^{54} & \mathbf{f}_5^{55} + \mathbf{f}_6^{55} & \mathbf{f}_6^{56} & 0 & 0 \\ \mathbf{f}_6^{63} & \mathbf{f}_6^{64} & \mathbf{f}_6^{65} & \mathbf{f}_6^{66} + \mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ 0 & 0 & 0 & \mathbf{f}_8^{76} & \mathbf{f}_7^{77} + \mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ 0 & 0 & 0 & \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix}. \tag{5.98}$$

The matrix $\boldsymbol{\mu}_{X_3\to f_6}$ is a summary of the second largest dashed box on the left in Fig. 5.1. Similarly as in (5.92)–(5.96), we obtain $\mathbf{J}^{(68)}$ from $\mathbf{J}^{(38)}$:

$$\mathbf{J}^{(68)} = \begin{bmatrix} \boldsymbol{\mu}_{f_6\to x_6} + \mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ \mathbf{f}_8^{76} & \mathbf{f}_7^{77} + \mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix}, \tag{5.99}$$

where

$$\boldsymbol{\mu}_{f_6\to X_6} \triangleq \left(\left(\begin{bmatrix} \boldsymbol{\mu}_{X_3\to f_6} + \mathbf{f}_6^{33} & \mathbf{f}_6^{34} & \mathbf{f}_6^{35} & \mathbf{f}_6^{36} \\ \mathbf{f}_6^{43} & \mathbf{f}_4^{44} + \mathbf{f}_6^{44} & \mathbf{f}_6^{45} & \mathbf{f}_6^{46} \\ \mathbf{f}_6^{53} & \mathbf{f}_6^{54} & \mathbf{f}_5^{55} + \mathbf{f}_6^{55} & \mathbf{f}_6^{56} \\ \mathbf{f}_6^{63} & \mathbf{f}_6^{64} & \mathbf{f}_6^{65} & \mathbf{f}_6^{66} \end{bmatrix}^{-1} \right)_{44} \right)^{-1}. \tag{5.100}$$

The matrix $\boldsymbol{\mu}_{f_6\to X_6}$ is a summary of the largest dashed box left in Fig. 5.1. Eventually, we obtain $\mathbf{J}^{(66)}$ from $\mathbf{J}^{(68)}$, again by means of the matrix inversion lemma:

$$\mathbf{J}^{(66)} = \boldsymbol{\mu}_{f_6\to X_6} + \boldsymbol{\mu}_{f_8\to X_6}, \tag{5.101}$$

where

$$\boldsymbol{\mu}_{f_8\to X_6} \triangleq \left(\left(\begin{bmatrix} \mathbf{f}_8^{66} & \mathbf{f}_8^{67} & \mathbf{f}_8^{68} \\ \mathbf{f}_8^{76} & \mathbf{f}_7^{77} + \mathbf{f}_8^{77} & \mathbf{f}_8^{78} \\ \mathbf{f}_8^{86} & \mathbf{f}_8^{87} & \mathbf{f}_8^{88} \end{bmatrix}^{-1} \right)_{11} \right)^{-1}. \tag{5.102}$$

The matrix $\boldsymbol{\mu}_{f_8\to X_6}$ summarizes the right box in Fig. 5.1. Since $\mathbf{J}^{(66)} \triangleq ([\mathbf{J}^{-1}]_{6:6})^{-1}$, it follows from (5.101)

$$[\mathbf{J}^{-1}]_{66} \triangleq [\mathbf{J}^{-1}]_{6:6} = (\boldsymbol{\mu}_{f_6\to X_6} + \boldsymbol{\mu}_{f_8\to X_6})^{-1}. \tag{5.103}$$

In conclusion, the updates (5.94)–(5.100) can be considered as "closing" the dashed boxes in Fig. 5.1. Eventually, $[\mathbf{J}^{-1}]_{66}$ is obtained from both summaries arriving at the edge X_6 as in (5.103). It is easy to verify that the other diagonal elements $[\mathbf{J}^{-1}]_{kk}$ can be computed similarly.

From this example, it is but a small step to the summary propagation algorithm for computing standard unconditional BCRBs. We consider the summaries as messages that are sent out of the corresponding box, as is illustrated in Fig. 5.2.

At nodes representing differentiable functions, messages are computed according to the following rule.

Standard unconditional BCRB Summary Rule:
The message out of the node $g(x_1, \ldots, x_\ell, y)$ (see Fig. 5.3(a)) along the edge X_ℓ is the matrix

$$\boldsymbol{\mu}_{g \to X_\ell} = \left(\left[(\mathbf{G} + \mathbf{M})^{-1} \right]_{\ell\ell} \right)^{-1}, \tag{5.104}$$

where

$$\mathbf{M} = \operatorname{diag}(\boldsymbol{\mu}_{X_1 \to g}, \ldots, \boldsymbol{\mu}_{X_{\ell-1} \to g}, 0) \tag{5.105}$$

$$\mathbf{G}_{ij} \overset{\triangle}{=} -\mathrm{E}_{XY}[\nabla_{x_i} \nabla_{x_j}^T \log g(X_1, \ldots, X_\ell, Y)] \tag{5.106}$$

$$= -\int_{x,y} p(x,y) \nabla_{x_i} \nabla_{x_j}^T \log g(x_1, \ldots, x_\ell, y) dx dy, \tag{5.107}$$

where it is assumed that the integrals (5.108) exist $\forall i$ and $j = 1, \ldots, \ell$.

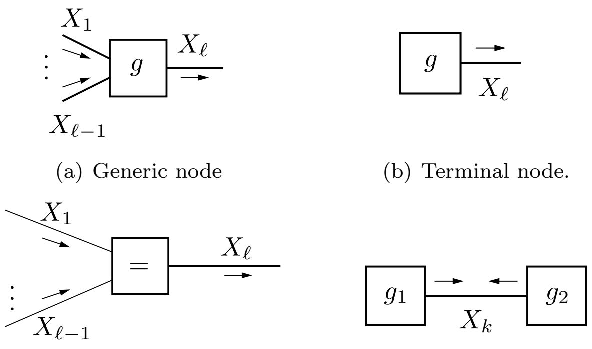
The expression (5.104) can be written in several ways; one can permute rows and corresponding columns of the matrices $\mathbf{M}$ and $\mathbf{G}$ (Lemma K.5). The expectations (5.108) can most often be simplified, since the local node g typically does not depend on the *whole* observation vector y, but on a small number of components y_k instead. Nevertheless, a closed-form expression for the expectations (5.108) may not exist; the expectations (5.108) may then be evaluated by numerical integration or by Monte Carlo methods. In the Monte-Carlo approach, one first draws a list of samples $\left\{ (\hat{x}^{(j)}, \hat{y}^{(j)}) \right\}_{j=1}^{N}$ from the joint pdf $p(x, y)$. Then, one evaluates (5.108) as an average over the samples $\left\{(\hat{x}^{(j)}, \hat{y}^{(j)})\right\}_{j=1}^{N}$:

$$\hat{\mathbf{G}}_{ij} \;\stackrel{\triangle}{=}\; -\sum_{j=1}^{N} \nabla_{x_i} \nabla_{x_j}^T \log g(x_1^{(j)}, \ldots, x_\ell^{(j)}, y^{(j)}). \tag{5.108}$$

It is usually easy to draw samples from $p(x, y)$; one may generate a sample $(\hat{x}, \hat{y})$ from $p(x, y)$ as follows:

a) Draw a sample $\hat{x}$ from $p(x)$,

b) Simulate the channel $p(y|x)$ with as input $\hat{x}$; this results in a sample $\hat{y}$ from $p(y|\hat{x})$.



(a) Generic node (b) Terminal node.

(c) Equality constraint node. (d) Edge X_k.

Figure 5.3: Summary propagation.

The message out of a terminal node $g(x_\ell, y)$ (see Fig. 5.3(b)) is defined as

$$\begin{aligned} \boldsymbol{\mu}_{g \to X_\ell} &\stackrel{\triangle}{=} -\mathrm{E}_{X_\ell Y}[\nabla_{x_\ell} \nabla_{x_\ell}^T \log g(X_\ell, Y)] && (5.109) \\ &= -\int_{x_\ell, y} p(x_\ell, y) \nabla_{x_\ell} \nabla_{x_\ell}^T \log g(x_\ell, y) dx_\ell dy. && (5.110) \end{aligned}$$

Half edges do not carry a message towards the (single) node attached to them; alternatively, they might be thought of as carrying a zero matrix as message. For the equality constraint node (see Fig. 5.3(c)), the integrals (5.108) do not exist, since the node "function" $f_=(x_1, x_2, \dots, x_\ell) \triangleq \delta(x_1 - x_2)\delta(x_2 - x_3)\dots\delta(x_{\ell-1} - x_\ell)$ is not differentiable. The equality constraint node has its own update rule; the outgoing message $\boldsymbol{\mu}_{\boxed{=}\to X_\ell}$ is the sum of the incoming messages $\boldsymbol{\mu}_{X_k\to\boxed{=}}$ $(k = 1, \dots, \ell - 1)$:

$$\boldsymbol{\mu}_{\boxed{=}\to X_\ell} = \sum_{k=1}^{\ell-1} \boldsymbol{\mu}_{X_k\to\boxed{=}}. \tag{5.111}$$

Deterministic nodes are handled by boxing (cf. Section 4.9.3). Eventually, the expression $[\mathbf{J}^{-1}]_{kk}$ is computed from the two messages $\boldsymbol{\mu}_{g_1\to X_k}$ and $\boldsymbol{\mu}_{g_2\to X_k}$ along the edge X_k (see Fig. 5.3(d)):

$$[\mathbf{J}^{-1}]_{kk} = (\boldsymbol{\mu}_{g_1\to X_k} + \boldsymbol{\mu}_{g_2\to X_k})^{-1}, \tag{5.112}$$

resulting in the bound:

$$\mathbf{E}_{kk} \succeq [\mathbf{J}^{-1}]_{kk} = (\boldsymbol{\mu}_{g_1\to X_k} + \boldsymbol{\mu}_{g_2\to X_k})^{-1}. \tag{5.113}$$

In the above, we have proposed a message-passing algorithm to compute standard unconditional Bayesian Cramér-Rao bounds. If one wishes to compute other Cramér-Rao-type bounds, one needs to slightly modify the algorithm. More precisely, the rules (5.104), (5.110) and (5.113) need to be adapted, the other rules remain unchanged. In the following, we explain how the rules (5.104) (5.110), and (5.113) should be modified for several other Cramér-Rao type bounds.

5.3.2 Conditional BCRBs

Suppose we wish to compute the conditional Bayesian Cramér-Rao bounds

$$\mathbf{E}_{kk}(y) \succeq [\mathbf{J}^{-1}(y)]_{kk}, \tag{5.114}$$

and

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk}(y) \succeq \sum_{k=1}^{n} w_k [\mathbf{J}^{-1}(y)]_{kk}. \tag{5.115}$$

The elements $[\mathbf{J}^{-1}(y)]_{kk}$ can be computed by the message-passing algorithm of Section 5.3.1, where the update rule (5.104) is replaced by the following rule.

Conditional BCRB Summary Rule:
The message out of the node $g(x_1, \ldots, x_\ell, y)$ (see Fig. 5.3(a)) along the edge X_ℓ is the matrix

$$\boldsymbol{\mu}_{g \to X_\ell} = \left(\left[(\mathbf{G} + \mathbf{M})^{-1} \right]_{\ell\ell} \right)^{-1}, \tag{5.116}$$

where

$$\mathbf{M} = \mathrm{diag}(\boldsymbol{\mu}_{X_1 \to g}, \ldots, \boldsymbol{\mu}_{X_{\ell-1} \to g}, 0) \tag{5.117}$$

$$\begin{aligned} \mathbf{G}_{ij} &\stackrel{\triangle}{=} -\mathrm{E}_{X|Y}[\nabla_{x_i} \nabla_{x_j}^T \log g(X_1, \ldots, X_\ell, y)] && (5.118) \\ &= -\int_x p(x|y) \nabla_{x_i} \nabla_{x_j}^T \log g(x_1, \ldots, x_\ell, y) dx, && (5.119) \end{aligned}$$

where it is assumed that the integrals (5.119) exist $\forall i$ and $j = 1, \ldots, \ell$.

In addition, the rules (5.110) and (5.113) need to be replaced by

$$\begin{aligned} \boldsymbol{\mu}_{g \to X_\ell} &\stackrel{\triangle}{=} -\mathrm{E}_{X_\ell|Y}[\nabla_{x_\ell} \nabla_{x_\ell}^T \log g(X_\ell, y)] && (5.120) \\ &= -\int_{x_\ell} p(x_\ell|y) \nabla_{x_\ell} \nabla_{x_\ell}^T \log g(x_\ell, y) dx_\ell, && (5.121) \end{aligned}$$

and

$$\mathbf{E}_{kk}(y) \succeq [\mathbf{J}^{-1}(y)]_{kk} = (\boldsymbol{\mu}_{g_1 \to X_k} + \boldsymbol{\mu}_{g_2 \to X_k})^{-1}. \tag{5.122}$$

respectively. Note that the expressions (5.119) and (5.121) involve averaging w.r.t. the posterior pdf $p(x|y)$, whereas (5.108) and (5.110) involve averaging over the joint pdf $p(x, y)$. If a closed-form expression for (5.119) and (5.121) is not available, one may resort to approximative methods such as numerical integration or Monte-Carlo methods. In the latter approach, the integrals in (5.119) and (5.121) are replaced by averages over a list of samples from the posterior $p(x|y)$. Note that it is usually substantially more difficult to sample from $p(x|y)$ than from $p(x, y)$.

5.3.3 Alternative Unconditional BCRBs

We consider now the alternative unconditional Bayesian Cramér-Rao bounds

$$\mathbf{E}_{kk} \succeq \mathrm{E}_Y \left[[\mathbf{J}^{-1}(Y)]_{kk} \right], \tag{5.123}$$

and

$$\sum_{k=1}^{n} w_k \mathbf{E}_{kk} \succeq \sum_{k=1}^{n} w_k \mathrm{E}_Y \left[[\mathbf{J}^{-1}(Y)]_{kk} \right]. \tag{5.124}$$

If the elements $\mathrm{E}_Y \left[[\mathbf{J}^{-1}(Y)]_{kk} \right]$ can not be computed analytically, they can be evaluated as an average over a list of samples $\left\{ \hat{y}^{(j)} \right\}_{j=1}^{N}$ from the pdf $p(y)$:

$$\mathrm{E}_Y \left[[\mathbf{J}^{-1}(Y)]_{kk} \right] = \sum_{j=1}^{N} [\mathbf{J}^{-1}(\hat{y}^{(j)})]_{kk}. \tag{5.125}$$

Each expression $[\mathbf{J}^{-1}(y^{(\ell)})]_{kk}$ may be determined by the message passing algorithm of Section 5.3.2.

5.3.4 Standard CRBs

Standard Cramér-Rao bounds for the MSE of a particular parameter θ_k can be computed by the message passing algorithm of Section 5.3.1, where the update rule (5.104) is replaced by the following rule.

Standard CRB Summary Rule:
The message out of the node $g(\theta_1, \ldots, \theta_\ell, y)$ (see Fig. 5.4) along the edge Θ_ℓ is the matrix

$$\boldsymbol{\mu}_{g \to \Theta_\ell} = \left(\left[(\mathbf{G} + \mathbf{M})^{-1} \right]_{\ell\ell} \right)^{-1}, \tag{5.126}$$

where

$$\mathbf{M} = \mathrm{diag}(\boldsymbol{\mu}_{\Theta_1 \to g}, \ldots, \boldsymbol{\mu}_{\Theta_{\ell-1} \to g}, 0) \tag{5.127}$$

$$\mathbf{G}_{ij} \triangleq -\mathrm{E}_{Y|\Theta}[\nabla_{\theta_i} \nabla_{\theta_j}^T \log g(\theta_1, \ldots, \theta_\ell, Y)] \tag{5.128}$$

$$= -\int_y p(y|\theta) \nabla_{\theta_i} \nabla_{\theta_j}^T \log g(\theta_1, \ldots, \theta_\ell, y) dy, \tag{5.129}$$

where it is assumed that the integrals (5.129) exist $\forall i$ and $j = 1, \ldots, \ell$.

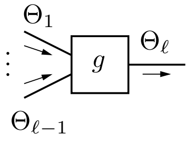


Figure 5.4: Generic node.

The rules (5.110) and (5.113) are replaced by:

$$\begin{aligned}\boldsymbol{\mu}_{g\to\Theta_\ell} &\stackrel{\triangle}{=} -\mathrm{E}_{Y|\Theta}[\nabla_{\theta_\ell}\nabla^T_{\theta_\ell}\log g(\theta_\ell, Y)] && (5.130)\\ &= -\int_y p(y|\theta)\nabla_{\theta_\ell}\nabla^T_{\theta_\ell}\log g(\theta_\ell, Y)dy, && (5.131)\end{aligned}$$

and

$$\mathbf{E}_{kk}(\theta) \succeq [\mathbf{F}^{-1}(\theta)]_{kk} = (\boldsymbol{\mu}_{g_1\to X_k} + \boldsymbol{\mu}_{g_2\to X_k})^{-1}. \qquad (5.132)$$

respectively. Note that (5.129) and (5.131) involve averaging w.r.t. the pdf $p(y|\theta)$. If the expressions (5.129) and (5.131) are intractable, they can be evaluated by numerical intregration or Monte Carlo integration. Note that it is usually easy to sample from the pdf $p(y|\theta)$.

We illustrate the above summary propagation algorithms by two simple examples.

Example 5.4. (Example 5.1 and 5.2 revisited)
We (re-)derive the (B)CRBs of Example 5.1 and 5.2 by *mechanically* applying the message-passing algorithms we described in the above. First, we derive the CRB (5.15), then, we derive the BCRBs (5.44), (5.45) and (5.46).

The CRB (5.15) may be computed by the summary-propagation procedure shown in Fig. 5.5. The messages $\boldsymbol{\mu}_{p\to\Theta_k}$ along the edges Θ_k are computed according to the rule (5.131), resulting in:

$$\boldsymbol{\mu}_{p\to\Theta_k} \stackrel{\triangle}{=} -\mathrm{E}_{Y|\Theta}\left[\frac{d^2}{d\theta^2}\log p(Y|\theta_k)\right] = 1/\sigma^2. \qquad (5.133)$$

The message $\boldsymbol{\mu}_{\boxminus\to\Theta}$ follows from the update rule (5.111) for equality

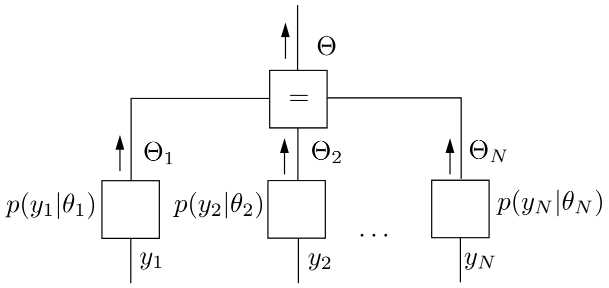


Figure 5.5: Computing the CRB (5.15) by message passing.

constraint nodes:

$$\begin{aligned}
\boldsymbol{\mu}_{\boxminus\to\Theta} &\triangleq \sum_{k=1}^{N} \boldsymbol{\mu}_{p\to\Theta_k} && (5.134)\\
&\triangleq -\sum_{k=1}^{N} \mathrm{E}_{Y|\Theta}\left[\frac{d^2}{d\theta^2}\log p(Y|\theta_k)\right] && (5.135)\\
&= N/\sigma^2. && (5.136)
\end{aligned}$$

The CRB follows from (5.132):

$$\mathbf{E}(\theta) \succeq [\mathbf{F}^{-1}(\theta)] = (\boldsymbol{\mu}_{\boxminus\to\Theta})^{-1} = \sigma^2/N. \qquad (5.137)$$

We now derive the standard unconditional BCRB (5.44) (see Fig. 5.6). The messages $\boldsymbol{\mu}_{p\to X_k}$ along the edges X_k are obtained from (5.110):

$$\boldsymbol{\mu}_{p\to X_k} \triangleq -\mathrm{E}_{X_k Y}\left[\frac{d^2}{dx_k^2}\log p(Y|X_k)\right] = 1/\sigma^2. \qquad (5.138)$$

The message $\boldsymbol{\mu}_{\boxminus\to X}$ is computed according to the update rule (5.111) for equality constraint nodes:

$$\begin{aligned}
\boldsymbol{\mu}_{\boxminus\to X} &\triangleq \sum_{k=1}^{N} \boldsymbol{\mu}_{p\to X_k} && (5.139)\\
&\triangleq -\sum_{k=1}^{N} \mathrm{E}_{X_k Y}\left[\frac{d^2}{dx^2}\log p(Y|X_k)\right] && (5.140)\\
&= N/\sigma^2. && (5.141)
\end{aligned}$$

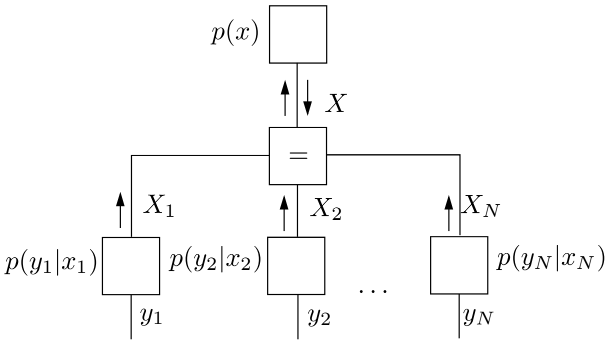


Figure 5.6: Computing the standard unconditional BCRB (5.44) by message passing.

The message $\boldsymbol{\mu}_{X\to\boxed{=}}$ follows from (5.110):

$$\boldsymbol{\mu}_{X\to\boxed{=}} = -\mathrm{E}_{XY}\left[\frac{d^2}{dx^2}\log p(X)\right] = -\mathrm{E}_X\left[\frac{d^2}{dx^2}\log p(X)\right]. \tag{5.142}$$

The standard unconditional BCRB for X follows from (5.113):

$$\mathbf{E} \succeq [\mathbf{J}^{-1}] \tag{5.143}$$
$$= \left(\boldsymbol{\mu}_{\boxed{=}\to X} + \boldsymbol{\mu}_{X\to\boxed{=}}\right)^{-1} \tag{5.144}$$
$$= \left(N/\sigma^2 - \mathrm{E}_X\left[\frac{d^2}{dx^2}\log p(X)\right]\right)^{-1}. \tag{5.145}$$

The BCRBs (5.45) and (5.46) are computed in a similar fashion. □

Example 5.5. ((B)CRBs for (unmodulated) constant phase)
We consider again the model:

$$Y_k \triangleq e^{j\Theta} + N_k, \tag{5.146}$$

where N_k is complex white Gaussian noise with (known) variance $2\sigma_N^2$, i.e., σ_N^2 per dimension, and $\Theta \in [0, 2\pi)$. We investigate the CRB for the problem of estimating the phase Θ from N observations $y_1, \ldots, y_N$.

The factor graph of Fig. 5.7 depicts the conditional pdf $p(y|\theta)$. The factors $p(y_k|\theta_k)$ are defined as

$$p(y_k|\theta_k) \triangleq \frac{1}{2\pi\sigma_N^2} e^{-|y_k - e^{j\theta_k}|^2/2\sigma_N^2}. \tag{5.147}$$

Since the factor graph of Fig. 5.7 is identical to the one of Fig. 5.5,

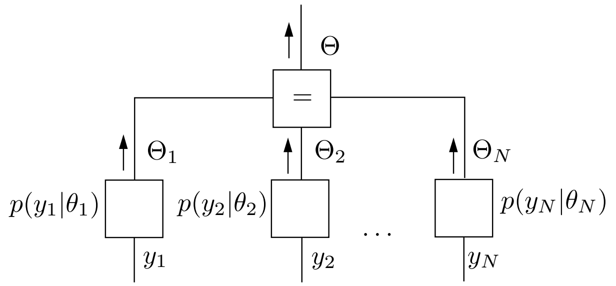


Figure 5.7: Computing the CRB for estimating a constant phase.

the CRB of the problem (5.146) is computed by the message-passing procedure we applied to obtain the CRB (5.15). The resulting CRB is identical to (5.15), as easily can be verified.

For the problem (5.146), the CRB (5.15) is *invalid* at finite SNR: Θ takes values in a finite interval, and all estimators are therefore biased. On the other hand, the CRB (5.15) is valid for all $\Theta \in (0, 2\pi)$ as $\sigma^2 \to 0$ or $N \to \infty$.

Note also that the MSE is not a suitable error measure if $\theta \neq \pi$, since the phase Θ is defined up to a multiple of 2π. A similar but suitable error measure is given by:

$$\mathcal{E}(\theta) \triangleq \int_y [s(\hat{\theta}(y) - \theta)]^2 p(y|\theta) dy \tag{5.148}$$

$$\triangleq \mathrm{E}_{Y|\Theta}\Big[s(\hat{\theta}(Y) - \theta)]^2\Big], \tag{5.149}$$

where θ and $\hat{\theta}(Y) \in [0, 2\pi)$, and $s(\cdot)$ is a periodic sawtooth function with period 2π, as depicted in Fig. 5.8.

If the MSE is not a suitable error measure for the estimation problem

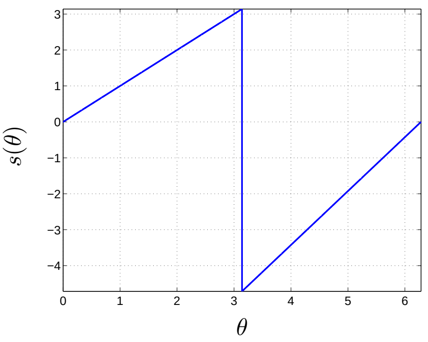


Figure 5.8: Periodic sawtooth function $s(\theta)$ with period 2π; one period is shown.

(5.146), is the CRB (5.15) relevant at all for the problem (5.146)? Fortunately, the answer is "yes". Note first of all that

$$\mathcal{E}(\pi) = \mathbf{E}(\pi) \triangleq \mathrm{E}_{Y|\Theta=\pi}\Big[\Big(\hat{\theta}(Y) - \pi\Big)^2\Big], \tag{5.150}$$

where $\hat{\theta}(Y) \in [0, 2\pi)$.

Moreover, due to symmetry,

$$\mathcal{E}(\theta) = \mathcal{E}(\pi), \quad \forall \theta \in [0, 2\pi). \tag{5.151}$$

As a consequence of (5.150) and (5.151),

$$\mathcal{E}(\theta) = \mathrm{E}_{Y|\Theta=\pi}\Big[\Big(\hat{\theta}(Y) - \pi\Big)^2\Big], \quad \forall \theta \in [0, 2\pi). \tag{5.152}$$

Therefore, the CRB (5.15), which also holds for $\Theta = \pi$, is a ("high-SNR") lower bound for $\mathcal{E}(\theta)$, for all $\theta \in [0, 2\pi)$. $\square$

5.3.5 Hybrid CRBs

Hybrid Cramér-Rao bounds for the MSE of a particular variable Z_k (random variable or parameter) can be computed by the message-passing algorithm of Section 5.3.1, where the update rule (5.104) is replaced by the following rule.

Hybrid CRB Summary Rule:
Consider the generic node $g(z_1, \ldots, z_\ell, y)$ (see Fig. 5.9), where the variable Z_k ($k = 1, 2 \ldots, \ell$) is a parameter (i.e., $Z_k \triangleq \Theta_k$) or a random variable (i.e., $Z_k \triangleq X_k$). The message out of the node along the edge Z_ℓ is the matrix

$$\boldsymbol{\mu}_{g \to Z_\ell} = \left(\left[(\mathbf{G} + \mathbf{M})^{-1}\right]_{\ell\ell}\right)^{-1}, \tag{5.153}$$

where

$$\mathbf{M} = \text{diag}(\boldsymbol{\mu}_{Z_1 \to g}, \ldots, \boldsymbol{\mu}_{Z_{\ell-1} \to g}, 0) \tag{5.154}$$

$$\mathbf{G}_{ij} \triangleq -\mathrm{E}_{XY|\Theta}[\nabla_{z_i} \nabla^T_{z_j} \log g(Z_1, \ldots, Z_\ell, Y)] \tag{5.155}$$

$$= -\int_{x,y} p(x, y|\theta) \nabla_{z_i} \nabla^T_{z_j} \log g(z_1, \ldots, z_\ell, y) dx dy, \tag{5.156}$$

where it is assumed that the integrals (5.156) exist $\forall i$ and $j = 1, \ldots, \ell$.

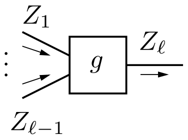


Figure 5.9: Generic node.

The rules (5.110) and (5.113) are replaced by

$$\boldsymbol{\mu}_{g \to Z_\ell} \triangleq -\mathrm{E}_{XY|\Theta}[\nabla_{z_\ell} \nabla^T_{z_\ell} \log g(Z_\ell, Y)]. \tag{5.157}$$

and

$$\mathrm{E}_{XY|\Theta}\left[(\hat{z}_k(Y) - Z_k)(\hat{z}_k(Y) - Z_k)^T\right] \succeq (\boldsymbol{\mu}_{g_1 \to Z_k} + \boldsymbol{\mu}_{g_2 \to Z_k})^{-1}. \tag{5.158}$$

respectively.

The expressions (5.156) and (5.157) involve averaging w.r.t. the pdf $p(x, y|\theta)$. If the expressions (5.156) and (5.157) are intractable, they can be evaluated by numerical intregration or Monte Carlo integration. Note that it is usually easy to sample from the pdf $p(x, y|\theta)$.

In the following sections, we compute standard unconditional BCRBs for "standard" estimation problems:

a) Filtering in state-space models (Section 5.3.6)

b) Smoothing in state-space models (Section 5.3.6)

c) Estimation of the parameters of state-space models (Section 5.3.7).

The extension to the other BCRBs is straightforward.

5.3.6 Estimation in State-Space Models

We consider a state-space model with freely evolving state X_k. The pdf $p(x, y)$ of such a system is given by

$$p(x, y) = p_0(x_0) \prod_{k=1}^{N} p(x_k|x_{k-1}) p(y_k|x_k), \tag{5.159}$$

its factor graph is shown in Fig. 5.14(a). Filtering corresponds to forward sum-product message passing through this factor graph. The standard unconditional BCRB for filtering is also computed in a forward sweep, as illustrated in Fig. 5.14(b) (ignore at this point the backward messages $\tilde{\mu}^B$ and μ^B); by applying the update rules (5.104) and (5.111) to the factor graph of Fig. 5.14(a), one obtains the recursion ($k = 0, \ldots, N-1$):

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}^F_{k+1} &= \left(\left[\left(\mathbf{G} + \operatorname{diag}(\boldsymbol{\mu}^F_k, 0)\right)^{-1}\right]_{22}\right)^{-1} && (5.160)\\
&= \left(\left(\begin{bmatrix} \boldsymbol{\mu}^F_k + \mathbf{G}_{k,11} & \mathbf{G}_{k,12} \\ \mathbf{G}_{k,21} & \mathbf{G}_{k,22} \end{bmatrix}^{-1}\right)_{22}\right)^{-1} && (5.161)\\
&= \mathbf{G}_{k,22} - \mathbf{G}_{k,21}(\boldsymbol{\mu}^F_k + \mathbf{G}_{k,11})^{-1}\mathbf{G}_{k,12} && (5.162)\\
\boldsymbol{\mu}^F_{k+1} &= \tilde{\boldsymbol{\mu}}^F_{k+1} + \boldsymbol{\mu}^Y_{k+1}, && (5.163)
\end{aligned}$$

where:

$$\begin{aligned}
\mathbf{G}_{k,11} &\triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla^T_{x_k} \log p(X_{k+1}|X_k)] && (5.164)\\
\mathbf{G}_{k,12} &\triangleq [\mathbf{G}_{k,21}]^T = \mathrm{E}_{XY}[-\nabla_{x_k}\nabla^T_{x_{k+1}} \log p(X_{k+1}|X_k)] && (5.165)\\
\mathbf{G}_{k,22} &\triangleq \mathrm{E}_{XY}[-\nabla_{x_{k+1}}\nabla^T_{x_{k+1}} \log p(X_{k+1}|X_k)] && (5.166)\\
\boldsymbol{\mu}^Y_k &\triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla^T_{x_k} \log p(Y_k|X_k)]. && (5.167)
\end{aligned}$$

The recursion is initialized by:

$$\boldsymbol{\mu}_0^F = \mathrm{E}_{X_0}[-\nabla_{x_0}\nabla_{x_0}^T \log p_0(X_0)]. \tag{5.168}$$

If the expectations (5.164)–(5.167) cannot be evaluated analytically, they can easily be evaluated by Monte-Carlo methods; indeed, in most applications, it is easy to sample from $p(x_k, x_{k+1})$ and $p(x_k, y_k)$. The standard unconditional BCRB for filtering is then ($k = 1, \ldots, N$):

$$\mathrm{E}_{XY}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] \overset{\triangle}{=} \mathbf{E}_{kk} \succeq \left(\boldsymbol{\mu}_k^F\right)^{-1}, \tag{5.169}$$

which was derived earlier in [192]. We have thus shown that the recursion of [192] can be viewed as forward-only message passing on the factor graph of (5.159). In the following, we derive the standard unconditional BCRB for smoothing, which to our knowledge is novel. Smoothing corresponds to updating messages according to the sum-product rule in a forward and a backward sweep [119]. Not surprisingly, the corresponding standard unconditional BCRB is also computed by a forward and backward sweep (Fig. 5.14(b)); the forward recursion is given by (5.160)–(5.168), the backward recursion for $\tilde{\boldsymbol{\mu}}_k^B$ and $\boldsymbol{\mu}_k^B$ is analogous. The backward recursion is initialized by $\tilde{\boldsymbol{\mu}}_N^B = 0$. The standard unconditional BCRB for smoothing is given by ($k = 1, \ldots, N$):

$$\mathbf{E}_{kk} \succeq \left(\tilde{\boldsymbol{\mu}}_k^F + \tilde{\boldsymbol{\mu}}_k^B + \boldsymbol{\mu}_k^Y\right)^{-1}. \tag{5.170}$$

Example 5.6. (BCRB for unmodulated time-variant phase)
We consider a channel of the form:

$$Y_k = e^{j\Theta_k} + N_k, \tag{5.171}$$

where N_k is an i.i.d. complex Gaussian random variable with (known) variance $2\sigma_N^2$, i.e., σ_N^2 per dimension. The evolution of the phase Θ_k is modelled as a random-walk process:

$$\Theta_k = (\Theta_{k-1} + W_k) \bmod 2\pi, \tag{5.172}$$

where W_k is a zero-mean (real) Gaussian random variable with known variance σ_W^2. The joint pdf of the system is given by

$$p(\theta, y) = p_0(\theta_0) \prod_{k=1}^{N} p(\theta_k|\theta_{k-1}) p(y_k|\theta_k), \tag{5.173}$$

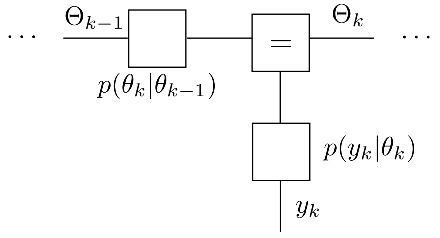


Figure 5.10: Factor graph of (5.173).

where

$$p(\theta_k|\theta_{k-1}) \stackrel{\triangle}{=} (2\pi\sigma_W^2)^{-1/2} \sum_{n\in\mathbf{Z}} e^{-((\theta_k-\theta_{k-1})+n2\pi)^2/2\sigma_W^2}, \tag{5.174}$$

and

$$p(y_k|z_k) \stackrel{\triangle}{=} (2\pi\sigma_N^2)^{-1}\ e^{-|y_k-z_k|^2/2\sigma_N^2}. \tag{5.175}$$

In the following, we will assume that $p_0(\theta_0) = 1/2\pi$ for all $\Theta_0 \in [0, 2\pi)$. A factor graph of the model (5.173) is shown in Fig. 5.10. The figure shows only one section ("time slice") of the graph; the total graph consists of many such sections, one for each time index k.

We investigate here the unconditional BCRB for the problem of estimating the phase Θ from N observations $y_1, \dots, y_N$. The system (5.173) is a state-space model with freely evolving state, hence, we can apply the update rules we derived earlier in this section. In this case,

$$\mathbf{G}_k^{kk} = \mathrm{E}_\Theta\Big[-\frac{d^2}{d\theta_k^2}\log p(\Theta_{k+1}|\Theta_k)\Big] \approx \frac{1}{\sigma_W^2} \tag{5.176}$$

$$\mathbf{G}_k^{k\,k+1} = -\mathbf{G}_k^{kk} \tag{5.177}$$

$$\mathbf{G}_k^{k+1\,k+1} = \mathbf{G}_k^{k\,k} \tag{5.178}$$

$$\boldsymbol{\mu}_k^Y = \mathrm{E}_{\Theta Y}\Big[-\frac{d^2}{d\theta_k^2}\log p(Y_{k+1}|\Theta_{k+1})\Big] = \frac{1}{\sigma_N^2}. \tag{5.179}$$

No closed-form expression exists for $\mathbf{G}_k^{kk}$; the approximation (5.176) is satisfactory as long as $\sigma_W^2 \leq 1$, as can be seen from Fig. 5.11. We obtain

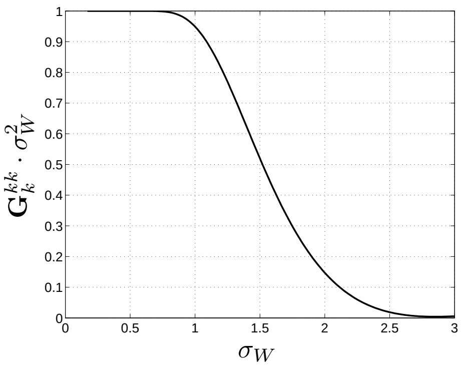


Figure 5.11: $\mathbf{G}_k^{kk} \cdot \sigma_W^2$ as a function of σ_W.

the forward and backward recursion

$$\tilde{\boldsymbol{\mu}}_k^F = \frac{1}{\sigma_W^2} - \frac{1}{\sigma_W^4}\left(\boldsymbol{\mu}_{k-1}^F + \frac{1}{\sigma_W^2}\right)^{-1} \tag{5.180}$$

$$\boldsymbol{\mu}_k^F = \tilde{\boldsymbol{\mu}}_k^F + \frac{1}{\sigma_N^2} \tag{5.181}$$

$$\tilde{\boldsymbol{\mu}}_k^B = \frac{1}{\sigma_W^2} - \frac{1}{\sigma_W^4}\left(\boldsymbol{\mu}_{k+1}^B + \frac{1}{\sigma_W^2}\right)^{-1} \tag{5.182}$$

$$\boldsymbol{\mu}_k^B = \tilde{\boldsymbol{\mu}}_k^B + \frac{1}{\sigma_N^2}. \tag{5.183}$$

The standard unconditional BCRB for filtering and smoothing is given by (5.169) and (5.170), where the involved messages are given by (5.180)–(5.183). In Fig. 5.12, those BCRBs are shown for particular values of σ_N^2 and σ_W^2. As can be seen from Fig. 5.12, the BCRB for filtering (forward sweep) decreases as k increases; the BCRB eventually converges to a steady-state value—as one would expect. The same holds for the BCRB of the backward sweep. The BCRB for smoothing attains the largest values at both ends of the block (i.e., for $k = 1$ and $k = N$); it evolves to a steady-state value towards the middle of the block. The standard unconditional BCRB for the MSE averaged over a block of length N= 100 is shown in Fig. 5.13. As expected, the BCRB decreases as the variances σ_N^2 and σ_W^2 decrease.

Note that the BCRBs are only valid as $\sigma_N^2 \to 0$, since the prior $p(\theta)$

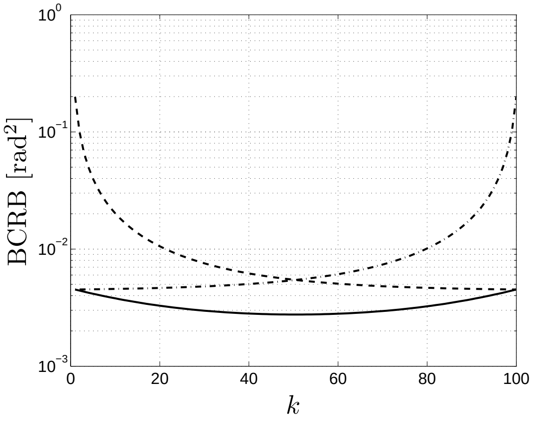


Figure 5.12: BCRBs for unmodulated random-walk phase model with $N = 100$, $\sigma_N = 0.446$ rad (4dB) and $\sigma_W^2 = 10^{-4}$ rad^2; Shown are the BCRB of the forward sweep, i.e., filtering (dashed line), BCRB of the backward sweep (dashed-dotted), and BCRB of smoothing (solid).

defined as:

$$p(\theta) \triangleq p_0(\theta_0) \prod_{k=1}^{N} p(\theta_k|\theta_{k-1}), \tag{5.184}$$

is non-zero at the boundary of its support.

□

So far, we have considered state-space models with *freely* evolving state. We now focus on *general* state-space models, i.e., input-driven state-space models. The pdf $p(u, x, y)$ of such a system is given by:

$$p(u, x, y) = p_0(x_0) \prod_{k=1}^{N} p(u_k, x_k, y_k|x_{k-1}), \tag{5.185}$$

where U is the input process. The factor graph of (5.185) is shown in Fig. 5.15(a), where p_k stands for the factor $p(u_k, x_k, y_k|x_{k-1})$, for $k = 1, 2, \ldots$ Applying the update rule (5.104) to this factor graph amounts

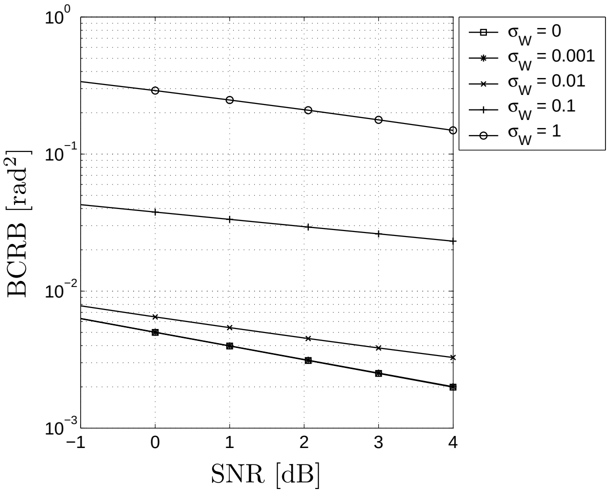


Figure 5.13: BCRB for the MSE averaged over a block of length $N = 100$, with $\sigma_W^2 = 0, 10^{-6}, 10^{-4}, 10^{-2}$, and 1 rad^2.

to the forward recursion (see Fig. 5.15(b)) ($k = 0, \ldots, N-1$):

$$\boldsymbol{\mu}_{k+1}^F = \left(\left[\left(\mathbf{G} + \operatorname{diag}(\boldsymbol{\mu}_k^F, 0, 0) \right)^{-1} \right]_{22} \right)^{-1} \tag{5.186}$$

$$= \left(\left(\begin{bmatrix} \boldsymbol{\mu}_k^F + \mathbf{G}_{k,11} & \mathbf{G}_{k,12} & \mathbf{G}_{k,13} \\ \mathbf{G}_{k,21} & \mathbf{G}_{k,22} & \mathbf{G}_{k,23} \\ \mathbf{G}_{k,31} & \mathbf{G}_{k,32} & \mathbf{G}_{k,33} \end{bmatrix}^{-1} \right)_{22} \right)^{-1} \tag{5.187}$$

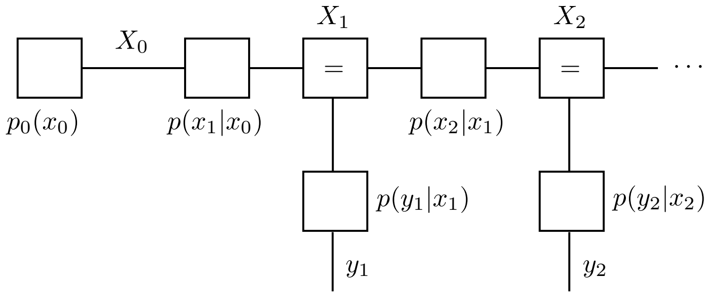
and a similar backward recursion ($k = 0, \ldots, N-1$):

$$\boldsymbol{\mu}_k^B = \left(\left[\left(\mathbf{G} + \operatorname{diag}(0, \boldsymbol{\mu}_{k+1}^B, 0) \right)^{-1} \right]_{11} \right)^{-1} \tag{5.188}$$

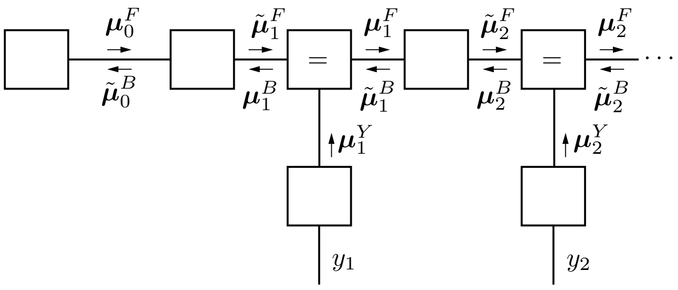
$$= \left(\left(\begin{bmatrix} \mathbf{G}_{k,11} & \mathbf{G}_{k,12} & \mathbf{G}_{k,13} \\ \mathbf{G}_{k,21} & \mathbf{G}_{k,22} + \boldsymbol{\mu}_{k+1}^B & \mathbf{G}_{k,23} \\ \mathbf{G}_{k,31} & \mathbf{G}_{k,32} & \mathbf{G}_{k,33} \end{bmatrix}^{-1} \right)_{11} \right)^{-1}, \tag{5.189}$$

where:

$$\mathbf{G}_{k,11} \stackrel{\triangle}{=} \mathrm{E}_{UXY}[-\nabla_{x_k} \nabla_{x_k}^T \log p(U_{k+1}, Y_{k+1}, X_{k+1} | X_k)] \tag{5.190}$$



(a) Factor graph.



(b) Summary propagation.

Figure 5.14: State space model with freely evolving state.

$$\mathbf{G}_{k,12} \triangleq \mathrm{E}_{UXY}[-\nabla_{x_k}\nabla^T_{x_{k+1}} \log p(U_{k+1}, Y_{k+1}, X_{k+1}|X_k)] \quad (5.191)$$

$$\mathbf{G}_{k,13} \triangleq \mathrm{E}_{UXY}[-\nabla_{x_k}\nabla^T_{u_{k+1}} \log p(U_{k+1}, Y_{k+1}, X_{k+1}|X_k)] \quad (5.192)$$

$$\mathbf{G}_{k,22} \triangleq \mathrm{E}_{UXY}[-\nabla_{x_{k+1}}\nabla^T_{x_{k+1}} \log p(U_{k+1}, Y_{k+1}, X_{k+1}|X_k)] \quad (5.193)$$

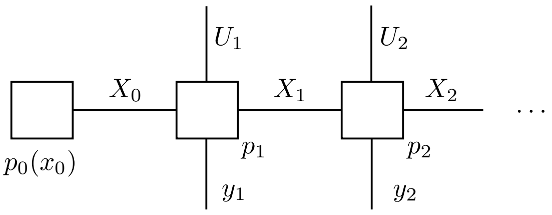
$$\mathbf{G}_{k,23} \triangleq \mathrm{E}_{UXY}[-\nabla_{x_{k+1}}\nabla^T_{u_{k+1}} \log p(U_{k+1}, Y_{k+1}, X_{k+1}|X_k)] \quad (5.194)$$

$$\mathbf{G}_{k,33} \triangleq \mathrm{E}_{UXY}[-\nabla_{u_{k+1}}\nabla^T_{u_{k+1}} \log p(U_{k+1}, Y_{k+1}, X_{k+1}|X_k)], \quad (5.195)$$

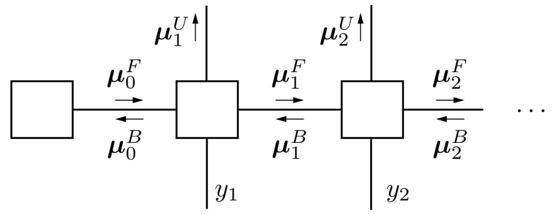
and $\mathbf{G}_{k,ij} = [\mathbf{G}_{k,ji}]^T$ for $i, j = 1, 2$ and 3. To compute the expectations (5.190)–(5.195), the joint pdf $p(u_{k+1}, x_k, x_{k+1}, y_{k+1})$ is required. It is usually straightforward to sample from $p(u_{k+1}, x_k, x_{k+1}, y_{k+1})$. Therefore, when a closed-form expression for the expectations (5.190)–(5.195) does not exist, they may be evaluated by Monte Carlo methods, as in state-space models with freely evolving state. The forward recursion is initialized by:

$$\boldsymbol{\mu}_0^F = \mathrm{E}_{X_0}[-\nabla_{x_0}\nabla^T_{x_0} \log p_0(X_0)]. \quad (5.196)$$

The backward recursion is initialized by $\boldsymbol{\mu}_N^B = 0$.



(a) Factor graph.


$\boldsymbol{\mu}_1^U\uparrow$
$\boldsymbol{\mu}_2^U\uparrow$
$\boldsymbol{\mu}_0^F \rightarrow$
$\boldsymbol{\mu}_1^F \rightarrow$
$\boldsymbol{\mu}_2^F \rightarrow$
$\leftarrow \boldsymbol{\mu}_0^B$
$\leftarrow \boldsymbol{\mu}_1^B$
$\leftarrow \boldsymbol{\mu}_2^B$
$\dots$
y_1
y_2

(b) Summary propagation.

Figure 5.15: General state-space model.

The standard unconditional BCRB for filtering X_k is again given by (5.169), where the messages $\boldsymbol{\mu}_k^F$ are now updated according to (5.186)–(5.187); the standard unconditional BCRB for smoothing X_k is of the form ($k = 1, \ldots, N$):

$$\mathrm{E}_{X_k Y}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] \succeq \left(\boldsymbol{\mu}_k^F + \boldsymbol{\mu}_k^B\right)^{-1}. \tag{5.197}$$

The messages $\boldsymbol{\mu}_k^U$ (see Fig. 5.15(b)) are computed from the messages $\boldsymbol{\mu}_k^F$ and $\boldsymbol{\mu}_k^B$ as ($k = 0, \ldots, N-1$):

$$\boldsymbol{\mu}_{k+1}^U = \left(\left[\left(\mathbf{G} + \mathrm{diag}(\boldsymbol{\mu}_k^F, \boldsymbol{\mu}_{k+1}^B, 0)\right)^{-1}\right]_{33}\right)^{-1} \tag{5.198}$$

$$= \left(\left(\begin{bmatrix} \mathbf{G}_{k,11} + \boldsymbol{\mu}_k^F & \mathbf{G}_{k,12} & \mathbf{G}_{k,13} \\ \mathbf{G}_{k,21} & \mathbf{G}_{k,22} + \boldsymbol{\mu}_{k+1}^B & \mathbf{G}_{k,23} \\ \mathbf{G}_{k,31} & \mathbf{G}_{k,32} & \mathbf{G}_{k,33} \end{bmatrix}^{-1}\right)_{33}\right)^{-1}. \tag{5.199}$$

The standard unconditional BCRB for filtering and smoothing the input U_k is given by ($k = 1, \ldots, N$):

$$\mathrm{E}_{U_k Y}[(\hat{u}_k(Y) - U_k)(\hat{u}_k(Y) - U_k)^T] \succeq \left[\boldsymbol{\mu}_k^U\right]^{-1}. \tag{5.200}$$

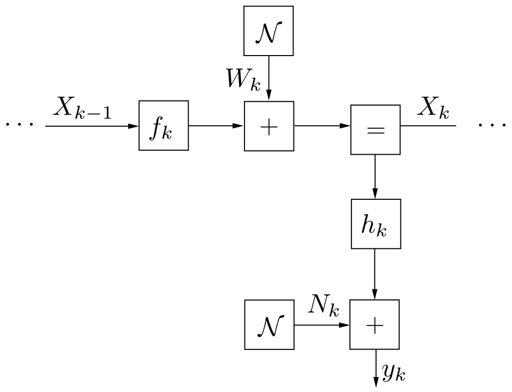
The message $\boldsymbol{\mu}_{k+1}^B$ in (5.199) is a zero matrix in the case of filtering; for smoothing, $\boldsymbol{\mu}_{k+1}^B$ is computed by the recursion (5.188)–(5.189).

Example 5.7. (BCRB for dynamical systems perturbed by additive Gaussian noise)
We consider the (non-linear) dynamical system:

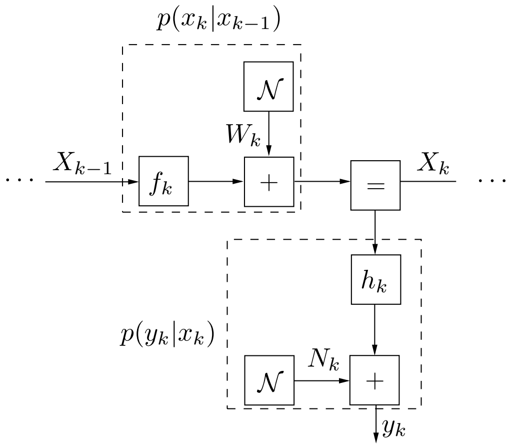
$$X_{k+1} = f_k(X_k) + W_k \tag{5.201}$$

$$Y_k = h_k(X_k) + N_k, \tag{5.202}$$

where $X_k \in \mathbb{R}^n$, $Y_k \in \mathbb{R}^m$, $f_k(\cdot)$ and $h_k(\cdot)$ are in general non-linear functions, and $W_k \in \mathbb{R}^n$ and $N_k \in \mathbb{R}^m$ are i.i.d. zero-mean Gaussian random vectors with (known) covariance matrices $\mathbf{Q}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{R}_k \in \mathbb{R}^{m\times m}$ respectively. A factor graph of (5.201)–(5.202) is depicted in Fig. 5.16(a). The figure shows only one section ("time slice") of the graph; the total graph consists of many such sections, one for each time index k. In the following, we compute the standard unconditional BCRB for estimating the state X_k from observations $y_1, \ldots, y_N$. We can not directly apply the BCRB summary-propagation algorithm to the factor graph of Fig. 5.16(a), since the graph contains deterministic nodes, i.e., two addition nodes and the (deterministic nodes) corresponding to f_k and h_k.



(a) Factor graph with deterministic nodes.



(b) Boxing.

Figure 5.16: Factor graph of (5.201)–(5.202).

We solve this problem by combining the deterministic nodes with continuous nodes ("boxing") as illustrated in Fig. 5.16(b). The upper dashed box stands for the factor:

$$\begin{aligned} p(x_k|x_{k-1}) &\triangleq \frac{1}{\sqrt{(2\pi)^n\,|\mathbf{Q}_k|}} e^{-1/2(x_k - f_k(x_{k-1}))^T \mathbf{Q}_k^{-1}(x_k - f_k(x_{k-1}))} && (5.203)\\ &= \mathcal{N}(x_k \mid f_k(x_{k-1}), \mathbf{Q}_k) && (5.204)\end{aligned}$$

the other dashed box stands for the factor:

$$\begin{aligned} p(y_k|x_k) &\triangleq \frac{1}{\sqrt{(2\pi)^m\,|\mathbf{R}_k|}} e^{-1/2(y_k - h_k(x_k))^T \mathbf{R}_k^{-1}(x_k - h_k(x_k))} && (5.205)\\ &= \mathcal{N}(y_k \mid h(x_k), \mathbf{R}_k)\,. && (5.206)\end{aligned}$$

By applying the update rules of Section 5.3.1 to the graph of Fig. 5.16(b), one obtains the forward recursion (5.160)–(5.163), and a similar backward recursion, where:

$$\begin{aligned} \mathbf{G}_{k,11} &\triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla_{x_k}^T \log p(X_{k+1}|X_k)] && (5.207)\\ &= \mathrm{E}_X[\mathbf{C}_{k+1}^T \mathbf{Q}_k^{-1}\mathbf{C}_{k+1}] && (5.208)\\ \mathbf{G}_{k,12} &\triangleq [\mathbf{G}_{k,21}]^T = \mathrm{E}_{XY}[-\nabla_{x_k}\nabla_{x_{k+1}}^T \log p(X_{k+1}|X_k)] && (5.209)\\ &= -\mathrm{E}_X[\mathbf{C}_{k+1}^T]\mathbf{Q}_k^{-1} && (5.210)\\ \mathbf{G}_{k,22} &\triangleq \mathrm{E}_{XY}[-\nabla_{x_{k+1}}\nabla_{x_{k+1}}^T \log p(X_{k+1}|X_k)] && (5.211)\\ &= \mathbf{Q}_k^{-1} && (5.212)\\ \boldsymbol{\mu}_k^Y &\triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla_{x_k}^T \log p(Y_k|X_k)] && (5.213)\\ &= \mathrm{E}_X[\mathbf{D}_{k+1}^T \mathbf{R}_{k+1}^{-1}\mathbf{D}_{k+1}], && (5.214)\end{aligned}$$

with

$$\begin{aligned} \mathbf{C}_{k+1}^T &= \nabla_{x_k} f_k^T(x_k) && (5.215)\\ \mathbf{D}_{k+1}^T &= \nabla_{x_{k+1}} h_{k+1}^T(x_{k+1}). && (5.216)\end{aligned}$$

The forward recursion (5.160)–(5.163) can be written as:

$$\begin{aligned} \boldsymbol{\mu}_{k+1}^F = \; & \mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1}\mathrm{E}_X[\mathbf{C}_{k+1}](\boldsymbol{\mu}_k^F + \mathrm{E}_X[\mathbf{C}_{k+1}^T\mathbf{Q}_k^{-1}\mathbf{C}_{k+1}])^{-1}\\ & \cdot \mathrm{E}_X[\mathbf{C}_{k+1}^T]\mathbf{Q}_k^{-1} + \mathrm{E}_X[\mathbf{D}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{D}_{k+1}]. \quad (5.217)\end{aligned}$$

The forward recursion (5.217) was derived earlier in [192]. The backward recursion is similar. The standard unconditional BCRB for filtering and smoothing is given by (5.169) and (5.170) respectively.

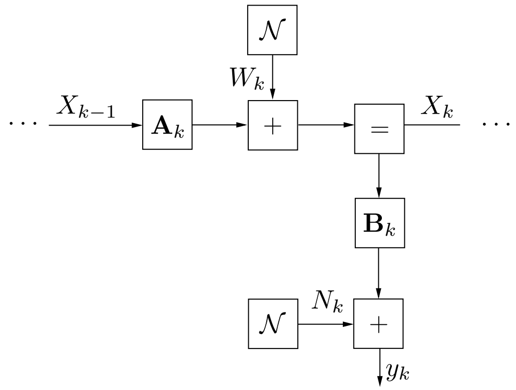


Figure 5.17: Factor graph of (5.218) (5.219).

We now consider the *linear* dynamical system[8] (see Fig. 5.17):

$$\begin{aligned} X_{k+1} &= \mathbf{A}_k X_k + W_k && (5.218) \\ Y_k &= \mathbf{B}_k X_k + N_k, && (5.219) \end{aligned}$$

where the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are known ("given"). Note that the two factors (5.205) and (5.206) are in this case Gaussian distributions in X and Y:

$$\begin{aligned} p(x_k|x_{k-1}) &\triangleq \frac{1}{\sqrt{(2\pi)^n\,|\mathbf{Q}_k|}} e^{-1/2(x_k-\mathbf{A}_k x_{k-1})^T \mathbf{Q}_k^{-1}(x_k-\mathbf{A}_k x_{k-1})} && (5.220) \\ p(y_k|x_k) &\triangleq \frac{1}{\sqrt{(2\pi)^m\,|\mathbf{R}_k|}} e^{-1/2(y_k-\mathbf{B}_k x_k)^T \mathbf{R}_k^{-1}(x_k-\mathbf{B}_k x_k)} && (5.221) \end{aligned}$$

Therefore, the joint pdf $p(x, y)$ is Gaussian.

The expressions (5.207)–(5.214) reduce to:

$$\begin{aligned} \mathbf{G}_{k,11} &\triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla_{x_k}^T \log p(X_{k+1}|X_k)] && (5.222) \\ &= \mathbf{A}_{k+1}^T \mathbf{Q}_k^{-1} \mathbf{C}_{k+1} && (5.223) \\ \mathbf{G}_{k,12} &\triangleq [\mathbf{G}_{k,21}]^T = \mathrm{E}_{XY}[-\nabla_{x_k}\nabla_{x_{k+1}}^T \log p(X_{k+1}|X_k)] && (5.224) \\ &= -\mathbf{B}_{k+1}^T \mathbf{Q}_k^{-1} && (5.225) \end{aligned}$$

[8]Estimation in linear dynamical systems is reviewed in Appendix H.

$$\mathbf{G}_{k,22} \triangleq \mathrm{E}_{XY}[-\nabla_{x_{k+1}}\nabla^T_{x_{k+1}} \log p(X_{k+1}|X_k)] \tag{5.226}$$
$$= \mathbf{Q}_k^{-1} \tag{5.227}$$
$$\boldsymbol{\mu}_k^Y \triangleq \mathrm{E}_{XY}[-\nabla_{x_k}\nabla^T_{x_k} \log p(Y_k|X_k)] \tag{5.228}$$
$$= \mathbf{B}^T_{k+1}\mathbf{R}^{-1}_{k+1}\mathbf{B}_{k+1}, \tag{5.229}$$

and the recursion (5.217) becomes:

$$\begin{aligned}\boldsymbol{\mu}^F_{k+1} \quad = \quad & \mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1}\mathbf{A}_{k+1}(\boldsymbol{\mu}^F_k + \mathbf{A}^T_{k+1}\mathbf{Q}_k^{-1}\mathbf{A}_{k+1})^{-1} \\ & \cdot\mathbf{A}^T_{k+1}\mathbf{Q}_k^{-1} + \mathbf{B}^T_{k+1}\mathbf{R}^{-1}_{k+1}\mathbf{B}_{k+1}.\end{aligned} \tag{5.230}$$

The recursion (5.230) may look familiar to the reader who has some background in Kalman filtering (see Appendix H). Indeed, the recursion (5.230) is nothing but the Kalman update rule for inverse covariance matrices. Table H.3 contains the Kalman update rules for two standard (compound) nodes; the update rules for the mean (m), covariance matrix (V) and inverse covariance matrix (W) are given. If one applies the update rules 5 and 6 for W to the graph of Fig. 5.17, one obtains (5.230). This does not come as a surprise: We underlined earlier that the joint pdf $p(x, y)$ of the system (5.218)–(5.219) is Gaussian; the standard unconditional BCRB (5.25) holds with equality if $p(x, y)$ is Gaussian, as we mentioned at the beginning of this chapter. □

5.3.7 Cyclic Graphical Models

So far, we have developed message passing algorithms for computing Cramér-Rao-type bounds for systems represented by *cycle-free* graphs. Many systems, however, are most naturally represented by *cyclic* factor graphs, e.g., coupled state-space models, which are ubiquitous in signal processing. Since the summary propagation algorithms of Section 5.3.1 to 5.3.5 perform local update rules, they could in principle also be applied to a cyclic graph. Because of the cycle(s), the algorithms would be iterative with no natural termination; in addition, one would not obtain the exact (B)CRBs. In order to obtain the exact (B)CRBs, one needs to transform the cyclic factor graph into a *cycle-free* graph, for example by clustering or stretching certain nodes and edges [103]. Computing (B)CRBs by applying the summary propagation algorithm on the resulting cycle-free graph is most often feasible, since the computational complexity scales cubically (not exponentially!) with the cluster size.

As an illustration, we compute the standard unconditional BCRBs for the problem of estimating the parameters of a state-space model, which is a standard problem in signal processing; we first consider constant parameters, then, time-variant parameters. For the model with constant parameters, we will derive the standard unconditional BCRBs *explicitly* by applying the matrix inversion lemma to the unconditional Bayesian information matrix. This will amount to message passing on a cycle-free factor graph obtained by clustering. For the model with time-variant parameters, we will list the message update rules for computing the standard unconditional BCRBs (without deriving them explicitly).

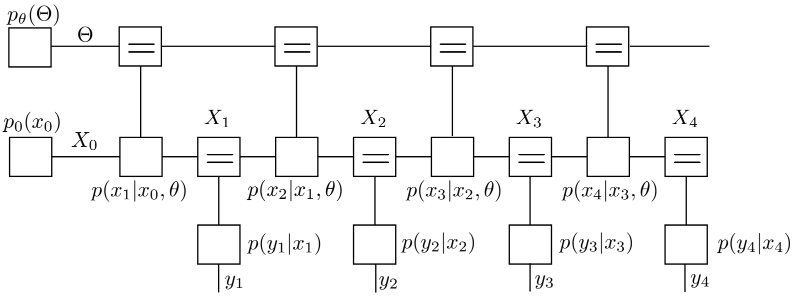
From both examples, it will become clear how the general update rule (5.104) for standard unconditional BCRBs (and likewise for the other (B)CRB update rules) can be extended to factor graphs in which some of the variables are represented by *several* edges, as is often the case after clustering or stretching [103].

State-Space Model with Constant Parameters

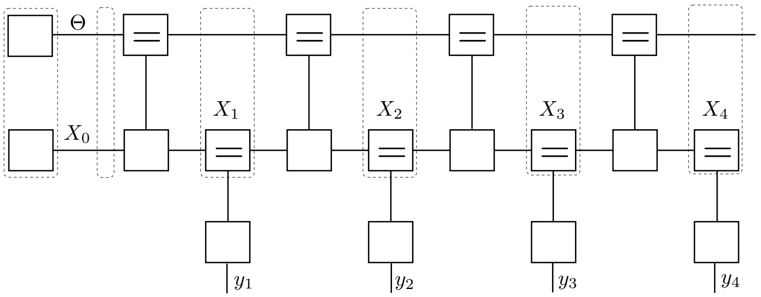
As a first example, we consider a state-space model whose transition probabilities $p(x_k|x_{k-1},\theta)$ are parametrized by an unknown *constant* parameter vector Θ with prior $p_\Theta(\theta)$. Such a system is in general described by the pdf

$$p(\theta, x, y) \triangleq p_\Theta(\theta) p_0(x_0) \prod_{k=1}^{N} p(x_k|x_{k-1},\theta) p(y_k|x_k). \tag{5.231}$$

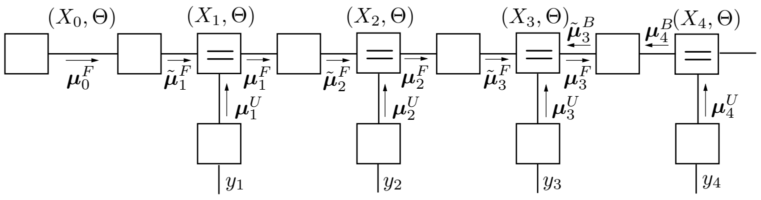
A factor graph that represents (5.231) with $N = 4$ is shown in Fig. 5.18(a). As an illustration, we compute the standard unconditional BCRB for the variable X_3 and Θ; the other variables X_k $(k = 0, 1, 2, 4)$ can be handled similarly. As in Section 5.3.1, we will determine the BCRBs by recursively applying the matrix inversion lemma (Lemma K.2). It will amount to BCRB summary propagation on the tree depicted in Fig. 5.18(c), obtained by clustering the state variables X_k and the parameter Θ, as shown in Fig. 5.18(b).



(a) Factor graph.



(b) Clustering.



(X_0,Θ)
(X_1,Θ)
(X_2,Θ)
(X_3,Θ)
(X_4,Θ)
$\overrightarrow{\boldsymbol{\mu}_0^F}$
$\overrightarrow{\tilde{\boldsymbol{\mu}}_1^F}$
$\overrightarrow{\boldsymbol{\mu}_1^F}$
$\overrightarrow{\tilde{\boldsymbol{\mu}}_2^F}$
$\overrightarrow{\boldsymbol{\mu}_2^F}$
$\overrightarrow{\tilde{\boldsymbol{\mu}}_3^F}$
$\overrightarrow{\boldsymbol{\mu}_3^F}$
$\overleftarrow{\tilde{\boldsymbol{\mu}}_3^B}$
$\overleftarrow{\boldsymbol{\mu}_4^B}$
$\boldsymbol{\mu}_1^U$
$\boldsymbol{\mu}_2^U$
$\boldsymbol{\mu}_3^U$
$\boldsymbol{\mu}_4^U$
y_1
y_2
y_3
y_4

(c) Summary propagation.

Figure 5.18: Estimation of (constant) parameters of a state-space model.

The unconditional Bayesian information matrix of (5.231) equals:

$$\mathbf{J}=\begin{bmatrix} \mathbf{f}_0^{00}+\mathbf{f}_1^{00} & \mathbf{f}_1^{01} & 0 & 0 & \mathbf{f}_1^{0\theta} & 0 \\ \mathbf{f}_1^{10} & \mathbf{y}_1^{11}+\mathbf{f}_1^{11}+\mathbf{f}_2^{11} & \mathbf{f}_2^{12} & 0 & \mathbf{f}_1^{1\theta}+\mathbf{f}_2^{1\theta} & 0 \\ 0 & \mathbf{f}_2^{21} & \mathbf{y}_2^{22}+\mathbf{f}_2^{22}+\mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \mathbf{f}_2^{2\theta}+\mathbf{f}_3^{2\theta} & 0 \\ 0 & 0 & \mathbf{f}_3^{32} & \mathbf{y}_3^{33}+\mathbf{f}_3^{33}+\mathbf{f}_4^{33} & \mathbf{f}_3^{3\theta}+\mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \mathbf{f}_1^{\theta 0} & \mathbf{f}_1^{\theta 1}+\mathbf{f}_2^{\theta 1} & \mathbf{f}_2^{\theta 2}+\mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3}+\mathbf{f}_4^{\theta 3} & \mathbf{f}_\theta^{\theta\theta}+\sum_{i=1}^4 \mathbf{f}_i^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ 0 & 0 & 0 & \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44}+\mathbf{f}_4^{44} \end{bmatrix}, \tag{5.232}$$

where

$$\mathbf{f}_0^{00} \triangleq -\mathrm{E}_X[\nabla_{x_0}\nabla_{x_0}^T \log p_0(X_0)], \tag{5.233}$$

$$\mathbf{f}_\theta^{\theta\theta} \triangleq -\mathrm{E}_\Theta[\nabla_\theta\nabla_\theta^T \log p_\Theta(\Theta)], \tag{5.234}$$

and for $k = 1, \ldots, 4$,

$$\mathbf{y}_k^{kk} \triangleq -\mathrm{E}_{XY\Theta}[\nabla_{x_k}\nabla_{x_k}^T \log p(Y_k|X_k)], \tag{5.235}$$

$$\mathbf{f}_k^{ij} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{x_i}\nabla_{x_j}^T \log p(X_k|X_{k-1},\Theta)], \tag{5.236}$$

$$\mathbf{f}_k^{i\theta} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{x_i}\nabla_\theta^T \log p(X_k|X_{k-1},\Theta)], \tag{5.237}$$

$$\mathbf{f}_k^{\theta i} \triangleq \left[\mathbf{f}_k^{i\theta}\right]^T, \tag{5.238}$$

$$\mathbf{f}_k^{\theta\theta} \triangleq -\mathrm{E}_{X\Theta}[\nabla_\theta\nabla_\theta^T \log p(X_k|X_{k-1},\Theta)]. \tag{5.239}$$

The first four rows and columns of $\mathbf{J}$ correspond to X_0 through X_3, whereas the fifth and the sixth row and column correspond to Θ and X_4 respectively. Obviously, one has the freedom to order the rows and columns in the Bayesian information matrix as one wishes. We have chosen this specific order to make the connection to BCRB summary propagation as clear as possible. From the standard unconditional BCRB (5.25) and the definition (5.232) of the unconditional Bayesian information matrix, it follows:

$$\mathrm{E}_{XY}[(\hat{x}_3(Y)-X_3)(\hat{x}_3(Y)-X_3)^T] \succeq [\mathbf{J}^{-1}]_{44} \tag{5.240}$$

$$\mathrm{E}_{\Theta Y}[(\hat{\theta}(Y)-\Theta)(\hat{\theta}(Y)-\Theta)^T] \succeq [\mathbf{J}^{-1}]_{55}. \tag{5.241}$$

In the following, we compute $[\mathbf{J}^{-1}]_{44}$ and $[\mathbf{J}^{-1}]_{55}$ by means of the matrix inversion lemma. First, we apply this lemma to $\mathbf{J}$ (cf. Lemma K.2

with $\mathbf{A} \triangleq \mathbf{J}$, $\mathbf{A}_{11} \triangleq \mathbf{J}_{1:1}$ and $\mathbf{A}_{22} \triangleq \mathbf{J}_{2:6}$), which leads to

$$\mathbf{J}^{(26)} \triangleq ([\mathbf{J}^{-1}]_{2:6})^{-1} \tag{5.242}$$

$$= \begin{bmatrix} \mathbf{y}_1^{11}+\mathbf{f}_1^{11}+\mathbf{f}_2^{11} & \mathbf{f}_2^{12} & 0 & \mathbf{f}_1^{1\theta}+\mathbf{f}_2^{1\theta} & 0 \\ \mathbf{f}_2^{21} & \mathbf{y}_2^{22}+\mathbf{f}_2^{22}+\mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \mathbf{f}_2^{2\theta}+\mathbf{f}_3^{2\theta} & 0 \\ 0 & \mathbf{f}_3^{32} & \mathbf{y}_3^{33}+\mathbf{f}_3^{33}+\mathbf{f}_4^{33} & \mathbf{f}_3^{3\theta}+\mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \mathbf{f}_1^{\theta 1}+\mathbf{f}_2^{\theta 1} & \mathbf{f}_2^{\theta 2}+\mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3}+\mathbf{f}_4^{\theta 3} & \mathbf{f}_\theta^{\theta\theta}+\sum_{i=1}^4 \mathbf{f}_i^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ 0 & 0 & \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44}+\mathbf{f}_4^{44} \end{bmatrix}$$

$$- \begin{bmatrix} \mathbf{f}_1^{10} \; 0 \; 0 \; 0 \; \mathbf{f}_1^{\theta 0} \end{bmatrix}^T \left(\mathbf{f}_0^{00} + \mathbf{f}_1^{00}\right)^{-1} \begin{bmatrix} \mathbf{f}_1^{10} \; 0 \; 0 \; 0 \; \mathbf{f}_1^{\theta 0} \end{bmatrix} \tag{5.243}$$

$$= \begin{bmatrix} \boldsymbol{\mu}_{1,11}^F+\mathbf{f}_2^{11} & \mathbf{f}_2^{12} & 0 & \boldsymbol{\mu}_{1,12}^F+\mathbf{f}_2^{1\theta} & 0 \\ \mathbf{f}_2^{21} & \mathbf{y}_2^{22}+\mathbf{f}_2^{22}+\mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \mathbf{f}_2^{2\theta}+\mathbf{f}_3^{2\theta} & 0 \\ 0 & \mathbf{f}_3^{32} & \mathbf{y}_3^{33}+\mathbf{f}_3^{33}+\mathbf{f}_4^{33} & \mathbf{f}_3^{3\theta}+\mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \boldsymbol{\mu}_{1,21}^F+\mathbf{f}_2^{\theta 1} & \mathbf{f}_2^{\theta 2}+\mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3}+\mathbf{f}_4^{\theta 3} & \boldsymbol{\mu}_{1,22}^F+\sum_{i=2}^4 \mathbf{f}_i^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ 0 & 0 & \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44}+\mathbf{f}_4^{44} \end{bmatrix},$$

where

$$\boldsymbol{\mu}_1^F \triangleq \tilde{\boldsymbol{\mu}}_1^F + \boldsymbol{\mu}_1^U, \tag{5.244}$$

with

$$\tilde{\boldsymbol{\mu}}_1^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{1,11}^F & \tilde{\boldsymbol{\mu}}_{1,12}^F \\ \tilde{\boldsymbol{\mu}}_{1,21}^F & \tilde{\boldsymbol{\mu}}_{1,22}^F \end{bmatrix} \tag{5.245}$$

$$\triangleq \left(\left(\begin{bmatrix} \mathbf{f}_0^{00}+\mathbf{f}_1^{00} & \mathbf{f}_1^{01} & \mathbf{f}_1^{0\theta} \\ \mathbf{f}_1^{10} & \mathbf{f}_1^{11} & \mathbf{f}_1^{1\theta} \\ \mathbf{f}_1^{\theta 0} & \mathbf{f}_1^{\theta 1} & \mathbf{f}_\theta^{\theta\theta}+\mathbf{f}_1^{\theta\theta} \end{bmatrix}^{-1} \right)_{2:3} \right)^{-1}, \tag{5.246}$$

and

$$\boldsymbol{\mu}_1^U \triangleq \begin{bmatrix} \mathbf{y}_1^{11} & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.247}$$

We now define

$$\boldsymbol{\mu}_0^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{0,11}^F & \boldsymbol{\mu}_{0,12}^F \\ \boldsymbol{\mu}_{0,21}^F & \boldsymbol{\mu}_{0,22}^F \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{f}_0^{00} & 0 \\ 0 & \mathbf{f}_\theta^{\theta\theta} \end{bmatrix}, \tag{5.248}$$

therefore

$$\tilde{\boldsymbol{\mu}}_1^F = \left(\left([\mathbf{M}_0^F + \mathbf{F}_1]^{-1} \right)_{2:3} \right)^{-1}, \tag{5.249}$$

where

$$\mathbf{M}_0^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{0,11}^F & 0 & \boldsymbol{\mu}_{0,12}^F \\ 0 & 0 & 0 \\ \boldsymbol{\mu}_{0,21}^F & 0 & \boldsymbol{\mu}_{0,22}^F \end{bmatrix}, \tag{5.250}$$

and

$$\mathbf{F}_1 \triangleq \begin{bmatrix} \mathbf{f}_1^{00} & \mathbf{f}_1^{01} & \mathbf{f}_1^{0\theta} \\ \mathbf{f}_1^{10} & \mathbf{f}_1^{11} & \mathbf{f}_1^{1\theta} \\ \mathbf{f}_1^{\theta 0} & \mathbf{f}_1^{\theta 1} & \mathbf{f}_1^{\theta\theta} \end{bmatrix}. \tag{5.251}$$

The matrix $\boldsymbol{\mu}_0^F$ is the summary of the left most dashed box in Fig. 5.18(b), whereas $\tilde{\boldsymbol{\mu}}_1^F$ summarizes the two left most dashed boxes. One obtains the message $\boldsymbol{\mu}_1^F$ from $\tilde{\boldsymbol{\mu}}_1^F$ by incorporating the observation y_1 according to (5.244). The latter is nothing but the BCRB message update rule (5.111) for equality constraint nodes, applied to the component X_1; the entries corresponding to Θ remain unchanged.

Along the same lines, one obtains

$$\mathbf{J}^{(36)} \triangleq \left([\mathbf{J}^{-1}]_{3:6}\right)^{-1} \tag{5.252}$$

$$= \begin{bmatrix} \mathbf{y}_2^{22} + \mathbf{f}_2^{22} + \mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \mathbf{f}_2^{2\theta} + \mathbf{f}_3^{2\theta} & 0 \\ \mathbf{f}_3^{32} & \mathbf{y}_3^{33} + \mathbf{f}_3^{33} + \mathbf{f}_4^{33} & \mathbf{f}_3^{3\theta} + \mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \mathbf{f}_2^{\theta 2} + \mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3} + \mathbf{f}_4^{\theta 3} & \boldsymbol{\mu}_{1,22}^F + \sum_{i=2}^4 \mathbf{f}_i^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ 0 & \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44} + \mathbf{f}_4^{44} \end{bmatrix} -$$

$$\begin{bmatrix} \mathbf{f}_2^{21} & 0 & 0 & \mathbf{f}_2^{\theta 1} \end{bmatrix}^T \left(\boldsymbol{\mu}_{1,11}^F + \mathbf{f}_2^{11}\right)^{-1} \begin{bmatrix} \mathbf{f}_2^{21} & 0 & 0 & \mathbf{f}_2^{\theta 1} \end{bmatrix} \tag{5.253}$$

$$= \begin{bmatrix} \boldsymbol{\mu}_{2,11}^F + \mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \boldsymbol{\mu}_{2,12}^F + \mathbf{f}_3^{2\theta} & 0 \\ \mathbf{f}_3^{32} & \mathbf{y}_3^{33} + \mathbf{f}_3^{33} + \mathbf{f}_4^{33} & \mathbf{f}_3^{3\theta} + \mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \boldsymbol{\mu}_{2,21}^F + \mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3} + \mathbf{f}_4^{\theta 3} & \boldsymbol{\mu}_{2,22}^F + \sum_{i=3}^4 \mathbf{f}_i^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ 0 & \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44} + \mathbf{f}_4^{44} \end{bmatrix}, \tag{5.254}$$

where

$$\boldsymbol{\mu}_2^F \triangleq \tilde{\boldsymbol{\mu}}_2^F + \boldsymbol{\mu}_2^U, \tag{5.255}$$

$$\tilde{\boldsymbol{\mu}}_2^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{2,11}^F & \tilde{\boldsymbol{\mu}}_{2,12}^F \\ \tilde{\boldsymbol{\mu}}_{2,21}^F & \tilde{\boldsymbol{\mu}}_{2,22}^F \end{bmatrix}$$

$$\triangleq \left(\left(\left[\mathbf{M}_1^F + \mathbf{F}_2\right]^{-1}\right)_{2:3}\right)^{-1}, \tag{5.256}$$

and

$$\boldsymbol{\mu}_2^U \triangleq \begin{bmatrix} \mathbf{y}_2^{22} & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.257}$$

with

$$\mathbf{M}_1^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{1,11}^F & 0 & \boldsymbol{\mu}_{1,12}^F \\ 0 & 0 & 0 \\ \boldsymbol{\mu}_{1,21}^F & 0 & \boldsymbol{\mu}_{1,22}^F \end{bmatrix}, \tag{5.258}$$

and

$$\mathbf{F}_2 \triangleq \begin{bmatrix} \mathbf{f}_2^{11} & \mathbf{f}_2^{12} & \mathbf{f}_2^{1\theta} \\ \mathbf{f}_2^{21} & \mathbf{f}_2^{22} & \mathbf{f}_2^{2\theta} \\ \mathbf{f}_2^{\theta 1} & \mathbf{f}_2^{\theta 2} & \mathbf{f}_2^{\theta\theta} \end{bmatrix}, \tag{5.259}$$

The matrices $\tilde{\boldsymbol{\mu}}_2^F$ and $\tilde{\boldsymbol{\mu}}_2^F$ can be interpreted as messages, as depicted in Fig. 5.18(c). The message $\boldsymbol{\mu}_2^F$ is computed from $\tilde{\boldsymbol{\mu}}_2^F$ by incorporating the observation y_2 according to (5.255), which is completely analogous to (5.244).

One obtains $\mathbf{J}^{(46)}$ from $\mathbf{J}^{(36)}$ as follows

$$\mathbf{J}^{(46)} \triangleq ([\mathbf{J}^{-1}]_{4:6})^{-1} \tag{5.260}$$

$$= \begin{bmatrix} \boldsymbol{\mu}_{3,11}^F + \mathbf{f}_4^{33} & \boldsymbol{\mu}_{3,12}^F + \mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \boldsymbol{\mu}_{3,21}^F + \mathbf{f}_4^{\theta 3} & \boldsymbol{\mu}_{3,22}^F + \mathbf{f}_4^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44} + \mathbf{f}_4^{44} \end{bmatrix} \tag{5.261}$$

where

$$\boldsymbol{\mu}_3^F \triangleq \tilde{\boldsymbol{\mu}}_3^F + \boldsymbol{\mu}_3^U, \tag{5.262}$$

$$\tilde{\boldsymbol{\mu}}_3^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{3,11}^F & \tilde{\boldsymbol{\mu}}_{3,12}^F \\ \tilde{\boldsymbol{\mu}}_{3,21}^F & \tilde{\boldsymbol{\mu}}_{3,22}^F \end{bmatrix} \tag{5.263}$$

$$\triangleq \left(\left([\mathbf{M}_2^F + \mathbf{F}_3]^{-1}\right)_{2:3}\right)^{-1}, \tag{5.264}$$

and

$$\boldsymbol{\mu}_3^U \triangleq \begin{bmatrix} \mathbf{y}_3^{33} & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.265}$$

with

$$\mathbf{M}_2^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{2,11}^F & 0 & \boldsymbol{\mu}_{2,12}^F \\ 0 & 0 & 0 \\ \boldsymbol{\mu}_{2,21}^F & 0 & \boldsymbol{\mu}_{2,22}^F \end{bmatrix}, \tag{5.266}$$

and

$$\mathbf{F}_3 \triangleq \begin{bmatrix} \mathbf{f}_3^{22} & \mathbf{f}_3^{23} & \mathbf{f}_3^{2\theta} \\ \mathbf{f}_3^{32} & \mathbf{f}_3^{33} & \mathbf{f}_3^{3\theta} \\ \mathbf{f}_3^{\theta 2} & \mathbf{f}_3^{\theta 3} & \mathbf{f}_3^{\theta\theta} \end{bmatrix}. \tag{5.267}$$

Similarly, $\mathbf{J}^{(45)}$ is given by

$$\mathbf{J}^{(45)} \triangleq ([\mathbf{J}^{-1}]_{4:5})^{-1} \tag{5.268}$$

$$\triangleq \boldsymbol{\mu}_3^{\text{tot}} \triangleq \boldsymbol{\mu}_3^F + \tilde{\boldsymbol{\mu}}_3^B \tag{5.269}$$

$$= \begin{bmatrix} \boldsymbol{\mu}_{3,11}^F + \tilde{\boldsymbol{\mu}}_{3,11}^B & \boldsymbol{\mu}_{3,12}^F + \tilde{\boldsymbol{\mu}}_{3,12}^B \\ \boldsymbol{\mu}_{3,21}^F + \tilde{\boldsymbol{\mu}}_{3,21}^B & \boldsymbol{\mu}_{3,22}^F + \tilde{\boldsymbol{\mu}}_{3,22}^B \end{bmatrix}, \tag{5.270}$$

where

$$\tilde{\boldsymbol{\mu}}_3^B \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{3,11}^B & \tilde{\boldsymbol{\mu}}_{3,12}^B \\ \tilde{\boldsymbol{\mu}}_{3,21}^B & \tilde{\boldsymbol{\mu}}_{3,22}^B \end{bmatrix} \tag{5.271}$$

$$\triangleq \left(\left(\begin{bmatrix} \mathbf{f}_4^{33} & \mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \mathbf{f}_4^{\theta 3} & \mathbf{f}_4^{\theta\theta} & \mathbf{f}_4^{\theta 4} \\ \mathbf{f}_4^{43} & \mathbf{f}_4^{4\theta} & \mathbf{y}_4^{44} + \mathbf{f}_4^{44} \end{bmatrix}^{-1} \right)_{1:2} \right)^{-1}. \tag{5.272}$$

We now define

$$\boldsymbol{\mu}_4^B \triangleq \begin{bmatrix} \boldsymbol{\mu}_{4,11}^B & \boldsymbol{\mu}_{4,12}^B \\ \boldsymbol{\mu}_{4,21}^B & \boldsymbol{\mu}_{4,22}^B \end{bmatrix} \triangleq \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{y}_4^{44} \end{bmatrix}. \tag{5.273}$$

As a consequence,

$$\tilde{\boldsymbol{\mu}}_3^B = \left(\left(\begin{bmatrix} \mathbf{f}_4^{33} & \mathbf{f}_4^{3\theta} & \mathbf{f}_4^{34} \\ \mathbf{f}_4^{\theta 3} & \boldsymbol{\mu}_{4,11}^B + \mathbf{f}_4^{\theta\theta} & \boldsymbol{\mu}_{4,12}^B + \mathbf{f}_4^{\theta 4} \\ \mathbf{f}_4^{43} & \boldsymbol{\mu}_{4,21}^B + \mathbf{f}_4^{4\theta} & \boldsymbol{\mu}_{4,22}^B + \mathbf{f}_4^{44} \end{bmatrix}^{-1} \right)_{1:2} \right)^{-1} \tag{5.274}$$

$$= \left(\left(\begin{bmatrix} \boldsymbol{\mu}^B_{4,11} + \mathbf{f}^{44}_4 & \mathbf{f}^{43}_4 & \boldsymbol{\mu}^B_{4,12} + \mathbf{f}^{4\theta}_4 \\ \mathbf{f}^{34}_4 & \mathbf{f}^{33}_4 & \mathbf{f}^{3\theta}_4 \\ \boldsymbol{\mu}^B_{4,21} + \mathbf{f}^{\theta 4}_4 & \mathbf{f}^{\theta 3}_4 & \boldsymbol{\mu}^B_{4,22} + \mathbf{f}^{\theta\theta}_4 \end{bmatrix}^{-1} \right)_{2:3} \right)^{-1} \qquad (5.275)$$

$$= \left(\left(\left[\mathbf{M}^B_4 + \mathbf{F}_4 \right]^{-1} \right)_{2:3} \right)^{-1}, \qquad (5.276)$$

with

$$\mathbf{M}^B_4 = \begin{bmatrix} \boldsymbol{\mu}^B_{4,11} & 0 & \boldsymbol{\mu}^B_{4,12} 0 \\ 0 & 0 & 0 \\ \boldsymbol{\mu}^B_{4,21} & 0 & \boldsymbol{\mu}^B_{4,22} \end{bmatrix}, \qquad (5.277)$$

and

$$\mathbf{F}_4 = \begin{bmatrix} \mathbf{f}^{33}_4 & \mathbf{f}^{3\theta}_4 & \mathbf{f}^{34}_4 \\ \mathbf{f}^{\theta 3}_4 & \mathbf{f}^{\theta\theta}_4 & \mathbf{f}^{\theta 4}_4 \\ \mathbf{f}^{43}_4 & \mathbf{f}^{4\theta}_4 & \mathbf{f}^{44}_4 \end{bmatrix}. \qquad (5.278)$$

The equality (5.275) follows from Lemma K.5. Note that the backward recursion (5.276) is analogous to the forward message update equations (5.249), (5.256), and (5.264).

Eventually, $\left[\mathbf{J}^{-1}\right]_{44}$ is obtained as

$$\left[\mathbf{J}^{-1}\right]_{44} = \left(\begin{bmatrix} \boldsymbol{\mu}^{\text{tot}}_{3,11} & \boldsymbol{\mu}^{\text{tot}}_{3,12} \\ \boldsymbol{\mu}^{\text{tot}}_{3,21} & \boldsymbol{\mu}^{\text{tot}}_{3,22} \end{bmatrix}^{-1} \right)_{11}. \qquad (5.279)$$

The diagonal elements $\left[\mathbf{J}^{-1}\right]_{kk}$ corresponding to the other coordinates X_k ($k = 1, 2, 3$) can be determined along the same lines. The diagonal element $\left[\mathbf{J}^{-1}\right]_{55}$, corresponding to Θ, is given by

$$\left[\mathbf{J}^{-1}\right]_{55} = \left(\begin{bmatrix} \boldsymbol{\mu}^{\text{tot}}_{3,11} & \boldsymbol{\mu}^{\text{tot}}_{3,12} \\ \boldsymbol{\mu}^{\text{tot}}_{3,21} & \boldsymbol{\mu}^{\text{tot}}_{3,22} \end{bmatrix}^{-1} \right)_{22}. \qquad (5.280)$$

In conclusion: the diagonal elements of the inverse unconditional Bayesian information matrix can be computed by message passing, as illustrated in Fig. 5.18(c). The messages are computed in a forward and backward sweep.

The forward messages are updated in two steps: first, the intermediate messages $\tilde{\boldsymbol{\mu}}_k^F$ are obtained from $\boldsymbol{\mu}_{k-1}^F$ (cf. (5.249), (5.256), and (5.264)) ($k = 1, \ldots, N$):

$$\tilde{\boldsymbol{\mu}}_k^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{k,11}^F & \tilde{\boldsymbol{\mu}}_{k,12}^F \\ \tilde{\boldsymbol{\mu}}_{k,21}^F & \tilde{\boldsymbol{\mu}}_{k,22}^F \end{bmatrix} \tag{5.281}$$

$$\triangleq \left(\left(\left[\mathbf{M}_{k-1}^F + \mathbf{F}_k\right]^{-1}\right)_{2:3}\right)^{-1}. \tag{5.282}$$

where

$$\mathbf{M}_{k-1}^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{k-1,11}^F & 0 & \boldsymbol{\mu}_{k-1,12}^F \\ 0 & 0 & 0 \\ \boldsymbol{\mu}_{k-1,21}^F & 0 & \boldsymbol{\mu}_{k-1,22}^F \end{bmatrix}, \tag{5.283}$$

and

$$\mathbf{F}_k \triangleq \begin{bmatrix} \mathbf{f}_k^{k-1\,k-1} & \mathbf{f}_k^{k-1\,k} & \mathbf{f}_k^{k-1\,\theta} \\ \mathbf{f}_k^{k\,k-1} & \mathbf{f}_k^{kk} & \mathbf{f}_k^{k\theta} \\ \mathbf{f}_k^{\theta\,k-1} & \mathbf{f}_k^{\theta k} & \mathbf{f}_k^{\theta\theta} \end{bmatrix}. \tag{5.284}$$

Next, the messages $\boldsymbol{\mu}_k^F$ are computed from $\tilde{\boldsymbol{\mu}}_k^F$ and $\boldsymbol{\mu}_k^U$ (cf. (5.244), (5.255), and (5.262)) ($k = 1, \ldots, N$):

$$\boldsymbol{\mu}_k^F \triangleq \tilde{\boldsymbol{\mu}}_k^F + \boldsymbol{\mu}_k^U, \tag{5.285}$$

where

$$\boldsymbol{\mu}_k^U \triangleq \begin{bmatrix} \mathbf{y}_k^{kk} & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.286}$$

The backward messages $\tilde{\boldsymbol{\mu}}_k^B$ and $\boldsymbol{\mu}_k^B$ ($k = 1, \ldots, N$) are determined likewise.

The forward recursion is initialized as:

$$\boldsymbol{\mu}_0^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{0,11}^F & \boldsymbol{\mu}_{0,12}^F \\ \boldsymbol{\mu}_{0,21}^F & \boldsymbol{\mu}_{0,22}^F \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{f}_0^{00} & 0 \\ 0 & \mathbf{f}_\theta^{\theta\theta} \end{bmatrix}, \tag{5.287}$$

the backward recursion is initialized as:

$$\tilde{\boldsymbol{\mu}}_N^B \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{N,11}^B & \tilde{\boldsymbol{\mu}}_{N,12}^B \\ \tilde{\boldsymbol{\mu}}_{N,21}^B & \tilde{\boldsymbol{\mu}}_{N,22}^B \end{bmatrix} \triangleq \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.288}$$

We define the messages $\boldsymbol{\mu}_k^{\text{tot}}$ as $(k = 1, \ldots, N)$:

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{tot}} &\stackrel{\triangle}{=} \tilde{\boldsymbol{\mu}}_k^F + \boldsymbol{\mu}_k^B && (5.289) \\ &= \boldsymbol{\mu}_k^F + \tilde{\boldsymbol{\mu}}_k^B. && (5.290) \end{aligned}$$

The standard unconditional BCRB for the variable X_k is then obtained as $(k = 1, \ldots, N)$:

$$\begin{aligned} &\mathrm{E}_{XY}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] && (5.291) \\ &\succeq \left(\left(\begin{bmatrix} \boldsymbol{\mu}_{k,11}^{\text{tot}} & \boldsymbol{\mu}_{k,12}^{\text{tot}} \\ \boldsymbol{\mu}_{k,21}^{\text{tot}} & \boldsymbol{\mu}_{k,22}^{\text{tot}} \end{bmatrix}^{-1} \right)_{11} \right)^{-1} && (5.292) \\ &= \boldsymbol{\mu}_{k,11}^{\text{tot}} - \left(\boldsymbol{\mu}_{k,21}^{\text{tot}}\right)^T \left(\boldsymbol{\mu}_{k,22}^{\text{tot}}\right)^{-1} \boldsymbol{\mu}_{k,12}^{\text{tot}}. && (5.293) \end{aligned}$$

Similarly, the standard unconditional BCRB for Θ is given by

$$\begin{aligned} &\mathrm{E}_{\Theta Y}[(\hat{\theta}(Y) - \Theta)(\hat{\theta}(Y) - \Theta)^T] && (5.294) \\ &\succeq \left(\left(\begin{bmatrix} \boldsymbol{\mu}_{k,11}^{\text{tot}} & \boldsymbol{\mu}_{k,12}^{\text{tot}} \\ \boldsymbol{\mu}_{k,21}^{\text{tot}} & \boldsymbol{\mu}_{k,22}^{\text{tot}} \end{bmatrix}^{-1} \right)_{22} \right)^{-1} && (5.295) \\ &= \boldsymbol{\mu}_{k,22}^{\text{tot}} - \left(\boldsymbol{\mu}_{k,12}^{\text{tot}}\right)^T \left(\boldsymbol{\mu}_{k,11}^{\text{tot}}\right)^{-1} \boldsymbol{\mu}_{k,21}^{\text{tot}}, && (5.296) \end{aligned}$$

where k is arbitrary, i.e., the standard unconditional BCRB for Θ can be obtained from any message $\boldsymbol{\mu}_k^{\text{tot}}$ $(k = 1, \ldots, N)$.

State-Space Model with Time-Dependent Parameters

As a second example, we again consider a state-space model whose transition probabilities $p(x_k|x_{k-1}, \theta_k)$ are parametrized by an unknown parameter vector Θ_k. Now, the parameter vector Θ_k is *time-variant*: it is described by a state-space model. The pdf of such a system is given by:

$$p(\theta, x, y) \stackrel{\triangle}{=} p_0(x_0) \left(\prod_{k=1}^{N} p(x_k|x_{k-1}, \theta_k) p(y_k|x_k) \right) p_1(\theta_1) \left(\prod_{k=2}^{N} p(\theta_k|\theta_{k-1}) \right), \tag{5.297}$$

which is depicted in Fig. 5.19(a). The system (5.297) consists of two (coupled) state-space models: one for X_k, the other for Θ_k.

The BCRBs can be computed by message passing on the tree shown in Fig. 5.19(c), which was obtained by clustering the state variables X_k with the parameters Θ_k (see Fig. 5.19(b)).

In the following, we will list the message update rules for the standard unconditional BCRBs; they can be derived by means of the matrix inversion lemma, as in the previous example. The extension to other BCRBs it straightforward.

The messages are updated in two steps; first, the (intermediate) messages $\tilde{\boldsymbol{\mu}}_k^F$ are obtained from $\boldsymbol{\mu}_{k-1}^F$ as $(k = 1, \ldots, N-1)$:

$$\tilde{\boldsymbol{\mu}}_k^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{k,11}^F & \tilde{\boldsymbol{\mu}}_{k,12}^F \\ \tilde{\boldsymbol{\mu}}_{k,21}^F & \tilde{\boldsymbol{\mu}}_{k,22}^F \end{bmatrix} \tag{5.298}$$

$$\triangleq \left(\left(\left[\mathbf{M}_{k-1}^F + \mathbf{F}_k \right]^{-1} \right)_{2:3} \right)^{-1}, \tag{5.299}$$

where

$$\mathbf{M}_{k-1}^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{k-1,11}^F & 0 & \boldsymbol{\mu}_{k-1,12}^F \\ 0 & 0 & 0 \\ \boldsymbol{\mu}_{k-1,21}^F & 0 & \boldsymbol{\mu}_{k-1,22}^F \end{bmatrix} \tag{5.300}$$

$$\mathbf{F}_k \triangleq \begin{bmatrix} \mathbf{f}_k^{k-1\,k-1} & \mathbf{f}_k^{k-1\,k} & \mathbf{f}_k^{k-1\,\hat{k}} \\ \mathbf{f}_k^{k\,k-1} & \mathbf{f}_k^{kk} & \mathbf{f}_k^{k\hat{k}} \\ \mathbf{f}_k^{\hat{k}\,k-1} & \mathbf{f}_k^{\hat{k}k} & \mathbf{f}_k^{\hat{k}\hat{k}} \end{bmatrix}, \tag{5.301}$$

and

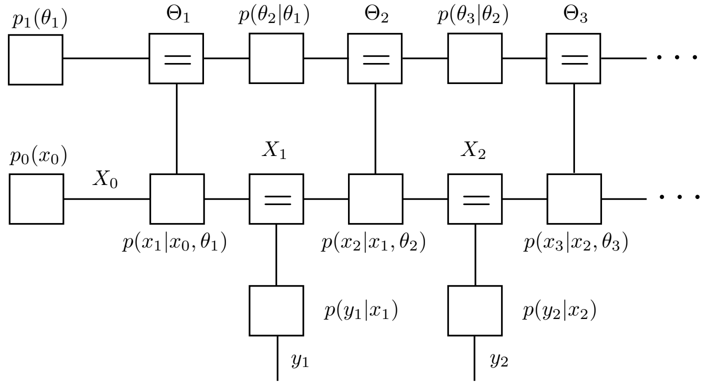
$$\mathbf{f}_k^{ij} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{x_i} \nabla_{x_j}^T \log p(X_k | X_{k-1}, \Theta_k)] \triangleq \left[\mathbf{f}_k^{ji} \right]^T \tag{5.302}$$

$$\mathbf{f}_k^{i\hat{k}} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{x_i} \nabla_{\theta_k}^T \log p(X_k | X_{k-1}, \Theta_k)] \triangleq \left[\mathbf{f}_k^{\hat{k}i} \right]^T \tag{5.303}$$

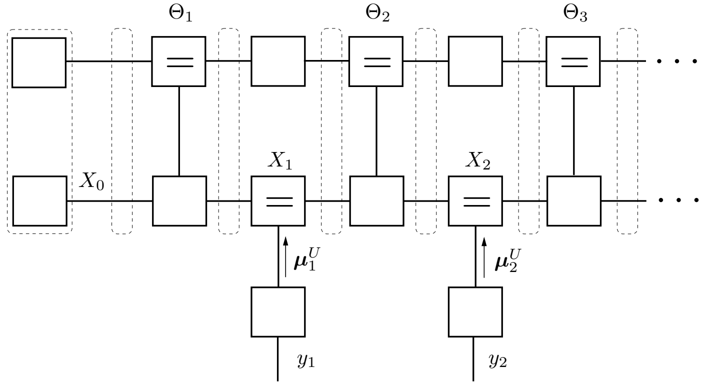
$$\mathbf{f}_k^{\hat{k}\hat{k}} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{\theta_k} \nabla_{\theta_k}^T \log p(X_k | X_{k-1}, \Theta_k)]. \tag{5.304}$$

Then, the messages $\boldsymbol{\mu}_k^F$ are computed from $\tilde{\boldsymbol{\mu}}_k^F$ as $(k = 1, \ldots, N)$:

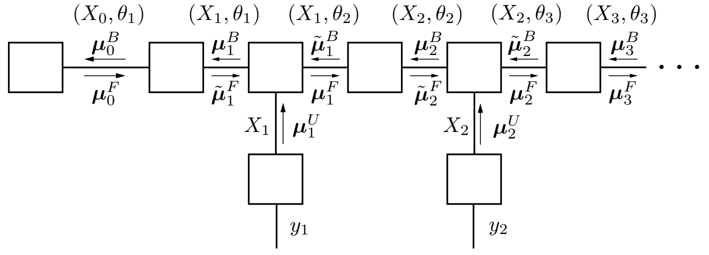
$$\boldsymbol{\mu}_k^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{k,11}^F & \boldsymbol{\mu}_{k,12}^F \\ \boldsymbol{\mu}_{k,21}^F & \boldsymbol{\mu}_{k,22}^F \end{bmatrix} \tag{5.305}$$



(a) Factor graph.



(b) Clustering.



(c) Summary propagation.

Figure 5.19: Estimation of (time-dependent) parameters of a state-space model.

$$\triangleq \left(\left(\left[\tilde{\mathbf{M}}_k^F + \mathbf{M}_k^U + \mathbf{G}_{k+1}\right]^{-1}\right)_{1:2}\right)^{-1}, \tag{5.306}$$

where

$$\tilde{\mathbf{M}}_k^F \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{k,11}^F & 0 & \tilde{\boldsymbol{\mu}}_{k,12}^F \\ 0 & 0 & 0 \\ \tilde{\boldsymbol{\mu}}_{k,21}^F & 0 & \tilde{\boldsymbol{\mu}}_{k,22}^F \end{bmatrix} \tag{5.307}$$

$$\mathbf{M}_k^U \triangleq \mathrm{diag}(\boldsymbol{\mu}_k^U, 0, 0) \tag{5.308}$$

$$\mathbf{G}_{k+1} \triangleq \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{g}_{k+1}^{\hat{k}+1\,\hat{k}+1} & \mathbf{g}_{k+1}^{\hat{k}+1\,\hat{k}} \\ 0 & \mathbf{g}_{k+1}^{\hat{k}\,\hat{k}+1} & \mathbf{g}_{k+1}^{\hat{k}\,\hat{k}} \end{bmatrix}, \tag{5.309}$$

and

$$\boldsymbol{\mu}_k^U \triangleq -\mathrm{E}_{XY}[\nabla_{x_k}\nabla_{x_k}^T \log p(Y_k|X_k)] \tag{5.310}$$

$$\mathbf{g}_k^{\hat{s}\hat{t}} \triangleq -\mathrm{E}_{X\Theta}[\nabla_{\theta_s}\nabla_{\theta_t}^T \log p(\Theta_k|\Theta_{k-1})] \triangleq \left[\mathbf{g}_k^{\hat{t}\hat{s}}\right]^T. \tag{5.311}$$

The backward messages $\tilde{\boldsymbol{\mu}}_k^B$ and $\boldsymbol{\mu}_k^B$ $(k = 1, \ldots, N)$ are updated similarly.

The forward recursion is initialized as:

$$\boldsymbol{\mu}_0^F \triangleq \begin{bmatrix} \boldsymbol{\mu}_{0,11}^F & \boldsymbol{\mu}_{0,12}^F \\ \boldsymbol{\mu}_{0,21}^F & \boldsymbol{\mu}_{0,22}^F \end{bmatrix} \tag{5.312}$$

$$\triangleq \begin{bmatrix} -\mathrm{E}_{X_0}[\nabla_{x_0}\nabla_{x_0}^T \log p_0(X_0)] & 0 \\ 0 & -\mathrm{E}_{\Theta_1}[\nabla_{\theta_1}\nabla_{\theta_1}^T \log p_1(\Theta_1)] \end{bmatrix}, \tag{5.313}$$

The backward recursion is initialized as

$$\tilde{\boldsymbol{\mu}}_N^B \triangleq \begin{bmatrix} \tilde{\boldsymbol{\mu}}_{N,11}^B & \tilde{\boldsymbol{\mu}}_{N,12}^B \\ \tilde{\boldsymbol{\mu}}_{N,21}^B & \tilde{\boldsymbol{\mu}}_{N,22}^B \end{bmatrix} \triangleq \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.314}$$

We define the messages $\boldsymbol{\mu}_k^{\mathrm{tot}}$ as $(k = 1, \ldots, N)$:

$$\boldsymbol{\mu}_k^{\mathrm{tot}} \triangleq \tilde{\boldsymbol{\mu}}_k^F + \boldsymbol{\mu}_k^B \tag{5.315}$$

$$= \boldsymbol{\mu}_k^F + \tilde{\boldsymbol{\mu}}_k^B. \tag{5.316}$$

The standard unconditional BCRB for the variable X_k is then obtained as $(k = 1, \ldots, N)$:

$$\mathrm{E}_{XY}[(\hat{x}_k(Y) - X_k)(\hat{x}_k(Y) - X_k)^T] \tag{5.317}$$

$$\succeq \left(\left(\begin{bmatrix} \boldsymbol{\mu}^{\mathrm{tot}}_{k,11} & \boldsymbol{\mu}^{\mathrm{tot}}_{k,12} \\ \boldsymbol{\mu}^{\mathrm{tot}}_{k,21} & \boldsymbol{\mu}^{\mathrm{tot}}_{k,22} \end{bmatrix}^{-1} \right)_{11} \right)^{-1} \tag{5.318}$$

$$= \boldsymbol{\mu}^{\mathrm{tot}}_{k,11} - \left(\boldsymbol{\mu}^{\mathrm{tot}}_{k,21}\right)^T \left(\boldsymbol{\mu}^{\mathrm{tot}}_{k,22}\right)^{-1} \boldsymbol{\mu}^{\mathrm{tot}}_{k,12}. \tag{5.319}$$

Similarly, the standard unconditional BCRB for the variable Θ_k is given by $(k = 1, \ldots, N)$:

$$\mathrm{E}_{\Theta Y}[(\hat{\theta}_k(Y) - \Theta_k)(\hat{\theta}_k(Y) - \Theta_k)^T] \tag{5.320}$$

$$\succeq \left(\left(\begin{bmatrix} \boldsymbol{\mu}^{\mathrm{tot}}_{k,11} & \boldsymbol{\mu}^{\mathrm{tot}}_{k,12} \\ \boldsymbol{\mu}^{\mathrm{tot}}_{k,21} & \boldsymbol{\mu}^{\mathrm{tot}}_{k,22} \end{bmatrix}^{-1} \right)_{22} \right)^{-1} \tag{5.321}$$

$$= \boldsymbol{\mu}^{\mathrm{tot}}_{k,22} - \left(\boldsymbol{\mu}^{\mathrm{tot}}_{k,12}\right)^T \left(\boldsymbol{\mu}^{\mathrm{tot}}_{k,11}\right)^{-1} \boldsymbol{\mu}^{\mathrm{tot}}_{k,21}. \tag{5.322}$$

5.3.8 Cramér-Rao-type Update Rules Revisited

Clustering and stretching typically amounts to graphs in which some of the variables are represented by *several* edges (e.g., Θ in Fig. 5.18(c) and Θ_k in Fig. 5.19(c)). The generic update rules of Section 5.3.1 to 5.3.5, however, are only applicable to factor graphs where each variable corresponds to one edge. From the previous two examples, we can learn how the update rule (5.104) (for standard unconditional BCRBs) can be extended: one simply needs to generalize the update rules (5.282), (5.299), and (5.306). In the following, we investigate how the rule (5.104) (for standard unconditional BCRBs) should be adapted; the update rules for the other (B)CRBs are modified similarly.

First, we need to introduce some definitions. Let S be a set of n variables, i.e., $S \triangleq \{x_1, x_2, \ldots, x_n\}$. Suppose $g(x_1, x_2, \ldots, x_n)$ is a real-valued function of these variables. In Fig. 5.20, a factor graph of g is shown, where S_k $(k = 1, \ldots, \ell)$ stands for a (non-empty) subset of S, such that $S = S_1 \cup S_2 \ldots \cup S_\ell$. Let $S_\ell \triangleq \{x_1, x_2, \ldots, x_m\}$ $(m \leq n)$, without any loss of generality. The messages $\boldsymbol{\mu}_k$ along the edges S_k are $N_k \times N_k$ block matrices, where N_k is the cardinality of S_k. Each row and column of the block matrix $\boldsymbol{\mu}_k$ corresponds to one variable in S_k.

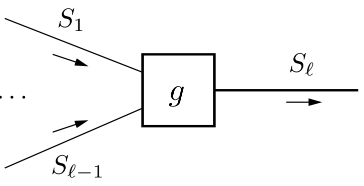


Figure 5.20: BCRB update rule.

We denote the submatrix of $\boldsymbol{\mu}_k$ at row s and column t by $\boldsymbol{\mu}_{k,st}$. We define the $n \times n$ block matrices $\mathbf{M}_k$ as ($k = 1, \dots, \ell - 1$):

$$\mathbf{M}_{k,ij} \triangleq \begin{cases} \boldsymbol{\mu}_{k,st} & \text{if row } s \text{ and column } t \text{ of } \boldsymbol{\mu}_k \text{ correspond to } x_i \text{ and } x_j, \\ 0 & \text{otherwise,} \end{cases} \tag{5.323}$$

where $\mathbf{M}_{k,ij}$ is the submatrix on the i-th row and j-th column of M_k. The i-th row and column of $\mathbf{M}_k$ correspond thus to X_i.

We define the matrix $\mathbf{G}$:

$$\mathbf{G} \triangleq \begin{bmatrix} \mathbf{G}_{11} & \dots & \mathbf{G}_{1\,n-1} & \mathbf{G}_{1n} \\ \vdots & \dots & \vdots & \vdots \\ \mathbf{G}_{n-1\,1} & \dots & \mathbf{G}_{n-1\,n-1} & \mathbf{G}_{n-1\,n} \\ \mathbf{G}_{n1} & \dots & \mathbf{G}_{n\,n-1} & \mathbf{G}_{nn} \end{bmatrix}, \tag{5.324}$$

where

$$\begin{aligned} \mathbf{G}_{ij} &\triangleq -\mathrm{E}_{XY}[\nabla_{x_i} \nabla_{x_j}^T \log g(X_1, \dots, X_n, Y)] & (5.325) \\ &= -\int_{x,y} p(x_1, \dots, x_n, y) \nabla_{x_i} \nabla_{x_j}^T \log g(x_1, \dots, x_n, y) dx dy, & (5.326) \end{aligned}$$

assuming this integral exist $\forall i$ and $j = 1, \dots, n$.

We are now able to state the generic update rule.

Extended standard unconditional BCRB Summary Rule:
The message $\boldsymbol{\mu}_n$, leaving the node g along the edge S_k, is computed as

$$\boldsymbol{\mu}_n = \left(\left((\mathbf{G} + \mathbf{M})^{-1}\right)_{1:m}\right)^{-1}, \tag{5.327}$$

where

$$\mathbf{M} \stackrel{\triangle}{=} \sum_{k=1}^{\ell-1} \mathbf{M}_k. \tag{5.328}$$

The expression (5.327) can be written in several ways; one can permute rows and corresponding columns of the matrices $\mathbf{M}_k$ and $\mathbf{G}$ (Lemma K.5).

We illustrate the rule (5.327) by a simple example.

Example 5.8. (Extended BCRB Summary Rule)
We derive the update rule (5.306) (for standard unconditional BCRBs) from (5.327) (see Fig. 5.19 and Fig. 5.21). In this case, $\ell = 3$, $S \stackrel{\triangle}{=} \{X_k, \Theta_k, \Theta_{k+1}\}$, $S_1 \stackrel{\triangle}{=} \{X_k, \Theta_k\}$, $S_2 \stackrel{\triangle}{=} \{X_{k+1}\}$, $S_3 \stackrel{\triangle}{=} \{X_k, \Theta_k\}$, $m = 2$, $N_1 = 2$, $N_2 = 1$, $N_3 = 2$, $X_1 \stackrel{\triangle}{=} X_k$, $X_2 \stackrel{\triangle}{=} \Theta_k$, $X_3 \stackrel{\triangle}{=} \Theta_{k+1}$, and

$$g(x_k, \theta_k, \theta_{k+1}) \stackrel{\triangle}{=} p(\theta_{k+1}|\theta_k). \tag{5.329}$$

The involved messages are:

$$\boldsymbol{\mu}_1 \stackrel{\triangle}{=} \tilde{\boldsymbol{\mu}}_k^F \tag{5.330}$$

$$\boldsymbol{\mu}_2 \stackrel{\triangle}{=} \boldsymbol{\mu}_k^U \tag{5.331}$$

$$\boldsymbol{\mu}_3 \stackrel{\triangle}{=} \boldsymbol{\mu}_k^F. \tag{5.332}$$

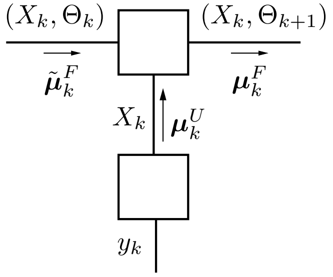
The matrices $\mathbf{M}_k$ (cf. (5.323)) equal:

$$\mathbf{M}_1 \stackrel{\triangle}{=} \tilde{\mathbf{M}}_k^F \tag{5.333}$$

$$\mathbf{M}_2 \stackrel{\triangle}{=} \mathbf{M}_k^U, \tag{5.334}$$

where $\mathbf{M}_k^F$ and $\mathbf{M}_k^U$ are given by (5.307) and (5.309) respectively. The matrix $\mathbf{G}$ (5.324) of the node function (5.329) is given by (5.309). With the previous definitions, the rule (5.327) boils down to (5.306).

□



(X_k, Θ_k)
(X_k, Θ_{k+1})
$\overrightarrow{\tilde{\boldsymbol{\mu}}}{}_k^F$
$\overrightarrow{\boldsymbol{\mu}}{}_k^F$
X_k
$\boldsymbol{\mu}_k^U$
y_k

Figure 5.21: BCRB update rule (5.327): example.

Example 5.9. (Hybrid CRBs for estimation in AR-models)
We consider the following problem.[9] Let $X_1, X_2, \ldots$ be a real random process ("auto-regressive (AR) model") defined by:

$$X_k = a_1 X_{n-1} + a_2 X_{k-2} + \cdots + a_M X_{k-M} + U_k, \tag{5.335}$$

where $a_1, \ldots, a_M$ are unknown real parameters, and $U_1, U_2, \ldots$ are real zero-mean Gaussian random variables with variance σ_U^2. We observe:

$$Y_k = X_k + W_k, \tag{5.336}$$

where W_k is (real) zero-mean white Gaussian noise with variance σ_W^2. We write (5.335) and (5.336) in state-space form as:

$$\mathbf{X}_k = \mathbf{A}\mathbf{X}_{n-1} + \mathbf{b}\, U_k \tag{5.337}$$

$$Y_k = \mathbf{c}^T \mathbf{X}_k + W_k, \tag{5.338}$$

with

$$\mathbf{X}_k \triangleq [X_k, \ldots, X_{k-M+1}]^T \tag{5.339}$$

$$\mathbf{A} \triangleq \left[\begin{array}{cc} \multicolumn{2}{c}{\mathbf{a}^T} \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right] \tag{5.340}$$

$$\mathbf{b} \triangleq \mathbf{c} \triangleq [1, 0, \ldots, 0]^T \tag{5.341}$$

$$\mathbf{a} \triangleq [a_1, \ldots, a_M]^T. \tag{5.342}$$

In (5.340), the matrix $\mathbf{I}$ is the $(M-1) \times (M-1)$ identity matrix and $\mathbf{0}$ is a zero vector of dimension $M-1$. The AR model (5.337)–(5.338) is

[9] Here, we will denote vectors in bold for convenience.

often used in signal processing, for example, to model speech signals or biomedical signals (e.g., EEG (ElectroEncephalogram) signals).

In the following, we consider the hybrid CRBs for the joint estimation of the state X_k, the coefficients $a_1, \ldots, a_M$, and the variances σ_U^2 and σ_W^2. More precisely, we consider the hybrid CRBs obtained from the information matrix of the *joint* pdf $p(x, y, u, w|\mathbf{a}, \sigma_W^2, \sigma_U^2)$. We will show that those hybrid CRBs are unfortunately loose. In Section 5.4.6, we will derive tighter Cramér-Rao-type bounds for this estimation problem, based on the information matrix of *marginals* of $p(x, y, u, w|\mathbf{a}, \sigma_W^2, \sigma_U^2)$.

The probability density function of the model (5.337)–(5.338) is given by:

$$\begin{aligned} & p(x, y, u, w|\mathbf{a}, \sigma_W^2, \sigma_U^2) \\ & \quad = p(\mathbf{x}_0) \prod_{k=1}^{N} p(\mathbf{x}_k|\mathbf{x}_{k-1}, u_k, \mathbf{a}) p(y_k|\mathbf{x}_k, w_k) p(u_k|\sigma_U^2) p(w_k|\sigma_W^2), \end{aligned} \tag{5.343}$$

where:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, u_k, \mathbf{a}) = \delta(\mathbf{x}_k - \mathbf{b}u_k - \mathbf{A}\mathbf{x}_{k-1}) \tag{5.344}$$

$$p(w_k|\sigma_W^2) = \mathcal{N}\big(w_k \mid 0, \sigma_W^2\big) \tag{5.345}$$

$$p(u_k|\sigma_U^2) = \mathcal{N}\big(u_k \mid 0, \sigma_U^2\big) \tag{5.346}$$

$$p(y_k|\mathbf{x}_k, w_k) = \delta(y_k - \mathbf{c}^T\mathbf{x}_k - w_k). \tag{5.347}$$

In the following, we will assume that no prior $p(\mathbf{x}_0)$ is specified (i.e., the prior $p(\mathbf{x}_0)$ is "uniform"). A factor graph for the system model (5.337)–(5.338) is depicted in Fig. 5.22; the figure shows only one section of the graph.

Note that the graph of Fig. 5.22 is cyclic and contains deterministic nodes, i.e., two addition nodes, and three matrix multiplication nodes. We need to convert the cyclic graph into a cycle-free graph before we can apply the summary propagation algorithm to compute the hybrid CRBs. In addition, we need to eliminate the deterministic nodes. By clustering and boxing, as shown in Fig. 5.23, we obtain the cycle-free graph of Fig. 5.24. The box $f_{1,k}$ stands for the factor ($k = M, \ldots, N$):

$$f_1(\mathbf{x}_k, \mathbf{x}_{k-1}, \mathbf{a}, \sigma_U^2) = \int_{u_k} \delta(\mathbf{x}_k - \mathbf{b}u_k - \mathbf{A}\mathbf{x}_{k-1}) \mathcal{N}\big(u_k \mid 0, \sigma_U^2\big)\, du_k \tag{5.348}$$

$$= \mathcal{N}_{0,\sigma_U^2}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1}) \tag{5.349}$$

$$= \mathcal{N}\left(x_k - \sum_{n=1}^{M} a_n x_{k-n} \mid 0, \sigma_U^2\right) \tag{5.350}$$

$$= f_1(x_k, \ldots, x_{k-M}, \mathbf{a}, \sigma_U^2), \tag{5.351}$$

whereas the box $f_{2,k}$ represents the factor $(k = 1, \ldots, N)$:

$$f_2(\mathbf{x}_k, y_k, \sigma_W^2) = \int_{w_k} \delta(y_k - \mathbf{c}^T\mathbf{x}_k - w_k)\mathcal{N}\big(w_k \mid 0, \sigma_W^2\big)\, dw_k \tag{5.352}$$

$$= \mathcal{N}\big(y_k - \mathbf{c}^T\mathbf{x}_k \mid 0, \sigma_W^2\big) \tag{5.353}$$

$$= \mathcal{N}\big(y_k - x_k \mid 0, \sigma_W^2\big) \tag{5.354}$$

$$\triangleq f_2(x_k, \sigma_W^2, y_k). \tag{5.355}$$

We are now ready to apply the (extended) hybrid CRB summary propagation algorithm to the graph of Fig. 5.23. As a first step, we determine the matrices $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ (cf. (5.324) and (5.327)), defined as:

$$\mathbf{G}_{ij}^{(1)} \triangleq -\mathrm{E}_{XY|\mathbf{a}\,\sigma_W^2\sigma_U^2}[\nabla_{z_i}\nabla_{z_j}^T \log f_1(X_k, \ldots, X_{k-M-1}, \mathbf{a}, \sigma_U^2)] \tag{5.356}$$

$$= -\int_{x,y} p(x, y|\mathbf{a}, \sigma_W^2, \sigma_U^2) \cdot\nabla_{z_i}\nabla_{z_j}^T \log f_1(x_k, \ldots, x_{k-M}, \mathbf{a}, \sigma_U^2)dxdy \tag{5.357}$$

$$\mathbf{G}_{ij}^{(2)} \triangleq -\mathrm{E}_{XY|\mathbf{a}\,\sigma_W^2\sigma_U^2}[\nabla_{z_i}\nabla_{z_j}^T \log f_2(X_k, \sigma_W^2, Y_k)] \tag{5.358}$$

$$= -\int_{z,y} p(x, y|\mathbf{a}, \sigma_W^2, \sigma_U^2)\nabla_{z_i}\nabla_{z_j}^T \log f_2(x_k, \sigma_W^2, y_k)dxdy, \tag{5.359}$$

with $Z_i \in \{X_k, \ldots, X_{k-M}, a_1, \ldots, a_M, \sigma_U^2, \sigma_W^2\}$.

After some straightforward algebra (see Appendix K), one obtains the

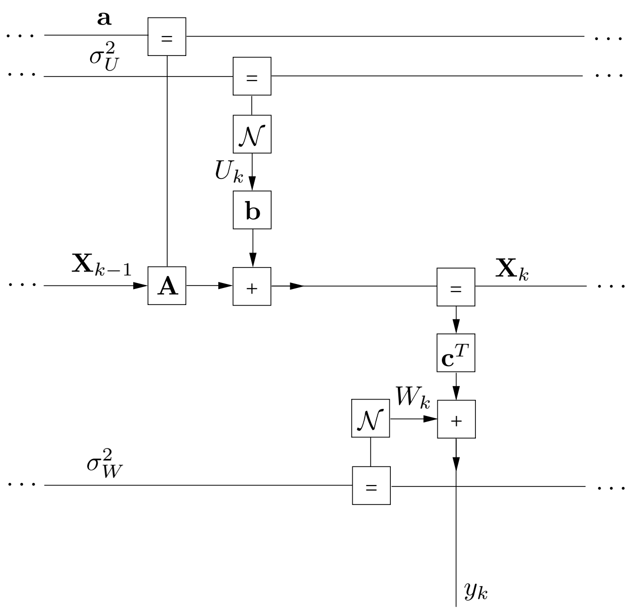


Figure 5.22: Factor graph of (5.337)–(5.338).

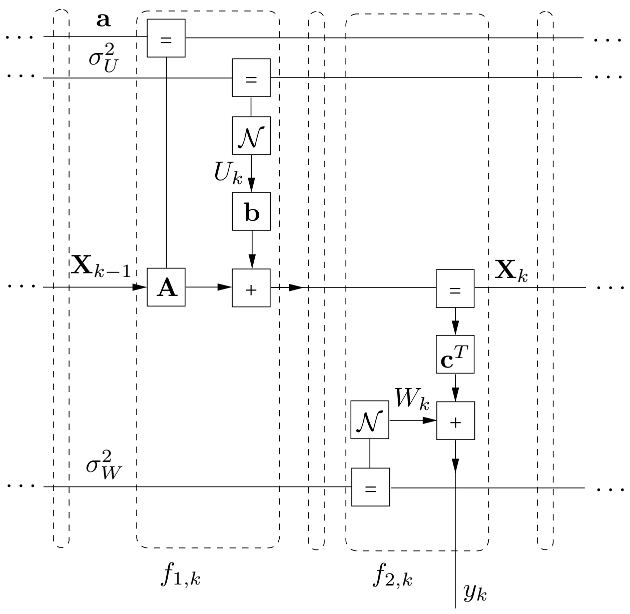


Figure 5.23: Clustering and boxing.

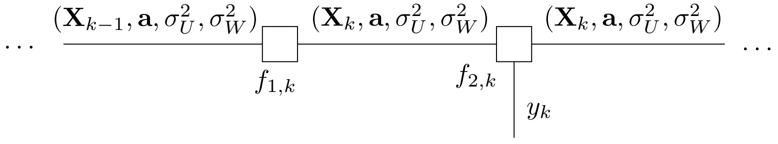

$(\mathbf{X}_{k-1}, \mathbf{a}, \sigma_U^2, \sigma_W^2)$
$(\mathbf{X}_k, \mathbf{a}, \sigma_U^2, \sigma_W^2)$
$(\mathbf{X}_k, \mathbf{a}, \sigma_U^2, \sigma_W^2)$
$f_{1,k}$
$f_{2,k}$
y_k

Figure 5.24: Tree representing the AR-model (5.337)–(5.338).

matrix $\mathbf{G}^{(1)}$:

$$\mathbf{G}^{(1)} = \left[\begin{array}{ccccc|cccc|c|c}
\frac{1}{\sigma_U^2} & -\frac{a_1}{\sigma_U^2} & -\frac{a_2}{\sigma_U^2} & \dots & -\frac{a_M}{\sigma_U^2} & 0 & 0 & \dots & 0 & 0 & 0 \\
-\frac{a_1}{\sigma_U^2} & \frac{a_1^2}{\sigma_U^2} & \frac{a_1 a_2}{\sigma_U^2} & \dots & \frac{a_1 a_M}{\sigma_U^2} & 0 & 0 & \dots & 0 & 0 & 0 \\
-\frac{a_2}{\sigma_U^2} & \frac{a_1 a_2}{\sigma_U^2} & \frac{a_2^2}{\sigma_U^2} & \dots & \frac{a_2 a_M}{\sigma_U^2} & 0 & 0 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\
-\frac{a_M}{\sigma_U^2} & \frac{a_1 a_M}{\sigma_U^2} & \frac{a_1 a_M}{\sigma_U^2} & \dots & \frac{a_M^2}{\sigma_U^2} & 0 & 0 & \dots & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \dots & 0 & \frac{C_{11}}{\sigma_U^2} & \frac{C_{12}}{\sigma_U^2} & \dots & \frac{C_{1M}}{\sigma_U^2} & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & \frac{C_{21}}{\sigma_U^2} & \frac{C_{22}}{\sigma_U^2} & \dots & \frac{C_{2M}}{\sigma_U^2} & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & \frac{C_{M1}}{\sigma_U^2} & \frac{C_{M2}}{\sigma_U^2} & \dots & \frac{C_{MM}}{\sigma_U^2} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2\sigma_U^4} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right], \tag{5.360}$$

where

$$C_{ij} \triangleq \mathrm{E}[X_{k-i}X_{k-j}]. \tag{5.361}$$

The matrix $\mathbf{G}^{(2)}$ equals:

$$\mathbf{G}^{(2)} = \left[\begin{array}{ccccc|cccc|c|c}
\frac{1}{\sigma_W^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2\sigma_W^4}
\end{array}\right]. \tag{5.362}$$

Given the matrices $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$, it is straightforward to derive the hybrid BCRBs:

$$\mathrm{E}\left[(\hat{\sigma}_U(Y) - \sigma_U)^2\right] \quad \geq \quad \frac{2\sigma_U^4}{N} \tag{5.363}$$

$$\mathrm{E}\left[(\hat{\sigma}_W(Y)-\sigma_W)^2\right] \geq \frac{2\sigma_W^4}{N-M} \tag{5.364}$$

$$\mathrm{E}\left[(\hat{a}_n(Y)-a_n)^2\right] \geq 0. \tag{5.365}$$

We leave the derivation of the hybrid BCRBs for X_k as an exercise for the reader. The resulting hybrid BCRBs are loose; this may directly be seen from the matrices $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$, i.e., even without computing the bounds explicitly. The off-diagonal submatrices of $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ are zero; this implies, for example, that the hybrid CRB for X_k is identical to the unconditional BCRB for X_k with *known* AR parameters $a_1, \ldots, a_M$, σ_U^2 and σ_W^2. In other words, the hybrid CRB for X_k does not take the fact into account that the AR parameters are *unknown*, hence, the bound is loose. The same holds for the hybrid CRBs for the AR parameters $a_1, \ldots, a_M$ and the variances σ_U^2 and σ_W^2; for example, the hybrid CRB for σ_U^2 does not depend on σ_W^2 and it does not take the fact into account that X is unkown. In Section 5.4.6 (cf. Example 5.10), we will derive tighter Cramér-Rao-type bounds for this estimation problem (from the information matrix of marginals). We will numerically evaluate those bounds together with the hybrid CRBs we have derived here. □

5.4 Cramér-Rao-Type Bounds From Marginal Densities

So far, we have derived Cramér-Rao-type bounds from the inverse information matrix of the *joint* pdf of the system at hand. In this section, we derive such bounds from inverse information matrices of *marginal* pdfs. The resulting bounds are often intractable; we present numerical algorithms to evaluate the bounds.

5.4.1 Standard CRB

We consider a system consisting of random variables X and parameters Θ. The joint pdf of the system is given by $p(x, y|\theta)$. We wish to compute the standard Cramér-Rao bound for Θ (cf. (5.80)):

$$\mathbf{E}_{11}^{(X\Theta)} \triangleq \mathrm{E}_{Y|\Theta}\Big[(\hat{\theta}(Y)-\theta)(\hat{\theta}(Y)-\theta)^T\Big]$$

$$\succeq \mathbf{F}^{-1}(\theta) \tag{5.366}$$

$$\stackrel{\triangle}{=} \mathrm{E}_{Y|\Theta}\Big[\nabla_\theta \log p(Y|\theta) \nabla_\theta^T \log p(Y|\theta)\Big]^{-1} \tag{5.367}$$

$$= -\mathrm{E}_{Y|\Theta}\Big[\nabla_\theta \nabla_\theta^T \log p(Y|\theta)\Big]^{-1}, \tag{5.368}$$

where

$$p(y|\theta) \stackrel{\triangle}{=} \int_x p(x,y|\theta)dx. \tag{5.369}$$

The following lemma paves the road to a numerical algorithm for computing the bound (5.366)–(5.368).

Lemma 5.3. If the integral $\int_x p(x,y|\theta)dx$ is differentiable under the integral sign (w.r.t. θ), then

$$\begin{aligned}&\mathrm{E}_{Y|\Theta}\Big[\nabla_\theta \log p(Y|\theta) \nabla_\theta^T \log p(Y|\theta)\Big]\\ &\quad = \mathrm{E}_{Y|\Theta}\Big[\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X,Y|\theta)\right] \mathrm{E}_{X|\Theta Y}[\nabla_\theta \log p(X,Y|\theta)]^T\Big].\end{aligned} \tag{5.370}$$

□

An equality similar to (5.370) has been proved earlier in the context of code-aided synchronization [149]. The expression in the RHS of (5.370) is usually as difficult to evaluate (analytically) as the expression in the LHS, in fact, both expressions are often intractable. One may then resort to *numerical* methods; the expression in the RHS of (5.370) suggests the following (numerical) algorithm to determine the bound (5.366)–(5.368):

a) Generate a list of samples $\{\hat{y}^{(j)}\}_{j=1}^N$ from $p(y|\theta)$.

b) Evaluate the expression:

$$\mathrm{E}_{X|\Theta Y}\Big[\nabla_\theta \log p(X,\hat{y}^{(j)}|\theta)\Big] \tag{5.371}$$

for $j = 1, \ldots, N$.

c) Compute the matrix $\hat{\mathbf{F}}^{(N)}(\theta)$:

$$\begin{aligned}\hat{\mathbf{F}}^{(N)}(\theta) \stackrel{\triangle}{=} \frac{1}{N}\sum_{j=1}^N \Bigg[&\mathrm{E}_{X|\Theta Y}\Big[\nabla_\theta \log p(X,\hat{y}^{(j)}|\theta)\Big]\\ &\cdot \mathrm{E}_{X|\Theta Y}\Big[\nabla_\theta \log p(X,\hat{y}^{(j)}|\theta)\Big]^T\Bigg].\end{aligned} \tag{5.372}$$

The matrix $\hat{\mathbf{F}}^{(N)}(\theta)$ is an approximation of the Fisher information matrix $\mathbf{F}(\theta)$. The approximation becomes more accurate as the number of samples N increases:

$$\lim_{N\to\infty} \hat{\mathbf{F}}^{(N)}(\theta) = \mathbf{F}(\theta). \tag{5.373}$$

Eventually, we replace the Fisher information matrix $\mathbf{F}(\theta)$ in (5.366)–(5.368) by the approximation $\hat{\mathbf{F}}^{(N)}(\theta)$:

$$\begin{aligned} \mathbf{E}_{11}^{(X\Theta)} &\triangleq \mathrm{E}_{Y|\Theta}\Big[(\hat{\theta}(Y)-\theta)(\hat{\theta}(Y)-\theta)^T\Big] \\ &\succeq \Big[\hat{\mathbf{F}}^{(N)}(\theta)\Big]^{-1}. \end{aligned} \tag{5.374}$$

Remarks:

- It is usually easy to sample from $p(y|\theta)$.
- The expression (5.371) is sometimes available in closed form (see, e.g., Example 5.10). If the expression (5.371) is intractable, one may use Monte-Carlo methods: the expression (5.371) is then evaluated as an average over a list of samples from $p(x|\theta, y^{(j)})$.

In the following, we list expressions similar to (5.370) for other Cramér-Rao-type bounds. They can be used to evaluate the (B)CRBs numerically, similarly as in (5.372) and (5.374).

5.4.2 Standard Unconditional BCRB

We now consider a system with random vectors X and Z; the joint pdf is given by $p(x, z, y)$. We wish to compute the standard unconditional BCRB for X (cf. (5.59)):

$$\begin{aligned} \mathbf{E} &\triangleq \mathrm{E}_{XY}[(\hat{x}(Y)-X)(\hat{x}(Y)-X)^T] \\ &\succeq \mathbf{J}^{-1} && (5.375) \\ &\triangleq \mathrm{E}_{XY}\left[\nabla_x \log p(X,Y)\nabla_x^T \log p(X,Y)\right]^{-1} && (5.376) \\ &= -\mathrm{E}_{XY}\big[\nabla_x\nabla_x^T \log p(X,Y)\big]^{-1}, && (5.377) \end{aligned}$$

where

$$p(x,y) \stackrel{\triangle}{=} \int_z p(x,z,y)dz. \tag{5.378}$$

An expression similar to (5.370) can be derived.

Lemma 5.4. If the integral $\int_z p(x,z,y)dz$ is differentiable under the integral sign (w.r.t. x), then

$$\begin{aligned} &\mathrm{E}_{XY}\big[\nabla_x \log p(X,Y)\nabla_x^T \log p(X,Y)\big] \\ &\quad = \mathrm{E}_{XY}\Big[\mathrm{E}_{Z|XY}[\nabla_x \log p(X,Z,Y)]\, \mathrm{E}_{Z|XY}[\nabla_x \log p(X,Z,Y)]^T\Big]. \end{aligned} \tag{5.379}$$

□

Lemma 5.4 amounts to the following algorithm to determine the bound (5.375)–(5.377):

a) Generate a list of samples $\{\hat{x}^{(j)}, \hat{y}^{(j)}\}_{j=1}^N$ from $p(x,y)$.

b) Evaluate the expression:

$$\mathrm{E}_{Z|XY}\Big[\nabla_x \log p(\hat{x}^{(j)}, Z, \hat{y}^{(j)})\Big] \tag{5.380}$$

for $j = 1, \ldots, N$.

c) Compute the matrix $\hat{\mathbf{J}}^{(N)}$:

$$\begin{aligned} \hat{\mathbf{J}}^{(N)} \stackrel{\triangle}{=} \frac{1}{N}\sum_{j=1}^{N}\Big[&\mathrm{E}_{Z|XY}\Big[\nabla_x \log p(\hat{x}^{(j)}, Z, \hat{y}^{(j)})\Big] \\ &\cdot \mathrm{E}_{Z|XY}\Big[\nabla_x \log p(\hat{x}^{(j)}, Z, \hat{y}^{(j)})\Big]^T\Big]. \end{aligned} \tag{5.381}$$

The bound (5.375)–(5.377) is eventually computed as:

$$\begin{aligned} \mathbf{E} &\stackrel{\triangle}{=} \mathrm{E}_{XY}[(\hat{x}(Y) - X)(\hat{x}(Y) - X)^T] \\ &\succeq \Big[\hat{\mathbf{J}}^{(N)}\Big]^{-1}. \end{aligned} \tag{5.382}$$

5.4.3 Conditional BCRB

We again consider a system with joint pdf $p(x, z, y)$. Now, we wish to compute the conditional BCRB for X:

$$\begin{aligned}
\mathbf{E}(y) &\stackrel{\triangle}{=} \mathrm{E}_{X|Y}[(\hat{x}(y) - X)(\hat{x}(y) - X)^T] \\
&\succeq \mathbf{J}^{-1}(y) && (5.383) \\
&\stackrel{\triangle}{=} \mathrm{E}_{X|Y}\left[\nabla_x \log p(X|y) \nabla_x^T \log p(X|y)\right]^{-1} && (5.384) \\
&= \mathrm{E}_{X|Y}\left[\nabla_x \log p(X,y) \nabla_x^T \log p(X,y)\right]^{-1} && (5.385) \\
&= -\mathrm{E}_{X|Y}\left[\nabla_x \nabla_x^T \log p(X|y)\right]^{-1} && (5.386) \\
&= -\mathrm{E}_{X|Y}\left[\nabla_x \nabla_x^T \log p(X,y)\right]^{-1}, && (5.387)
\end{aligned}$$

where

$$p(x|y) \stackrel{\triangle}{=} \int_z p(x,z|y)dz. \tag{5.388}$$

To this end, we slightly modify Lemma 5.4.

Lemma 5.5. If the integral $\int_z p(x, z, y)dz$ is differentiable under the integral sign (w.r.t. x), then

$$\begin{aligned}
&\mathrm{E}_{X|Y}\left[\nabla_x \log p(X,y) \nabla_x^T \log p(X,y)\right] \\
&\quad = \mathrm{E}_{X|Y}\left[\mathrm{E}_{Z|XY}\left[\nabla_x \log p(X,Z,y)\right] \mathrm{E}_{Z|XY}\left[\nabla_x \log p(X,Z,y)\right]^T\right].
\end{aligned} \tag{5.389}$$

□

A numerical algorithm for computing (5.383)–(5.387) follows directly from Lemma 5.5. The algorithm is analogous to the algorithm (5.380)–(5.382) for standard unconditional BCRBs. We leave the details as an exercise for the reader.

5.4.4 Alternative unconditional BCRB

Lemma 5.5 can also be used to compute the alternative unconditional BCRB:

$$\mathbf{E} \stackrel{\triangle}{=} \mathrm{E}_{XY}[(\hat{x}(y) - X)(\hat{x}(y) - X)^T]$$

$$\succeq \mathrm{E}_Y\left[\mathbf{J}^{-1}(y)\right] \tag{5.390}$$

$$\triangleq \mathrm{E}_Y\left[\mathrm{E}_{X|Y}\left[\nabla_x \log p(X,y)\nabla_x^T \log p(X,y)\right]^{-1}\right] \tag{5.391}$$

$$\succeq -\mathrm{E}_Y\left[\mathrm{E}_{X|Y}\left[\nabla_x\nabla_x^T \log p(X,y)\right]^{-1}\right]. \tag{5.392}$$

The expectation $\mathrm{E}_{X|Y}[\cdot]$ in (5.391) can be evaluated according to (5.389). The resulting algorithm for computing (5.390)–(5.392) is strikingly similar to the algorithm (5.380)–(5.382) for computing standard unconditional BCRB (5.375)–(5.377). Instead of the matrix $\hat{\mathbf{J}}^{(N)}$ (5.372), one computes the matrix $\hat{\mathbf{J}}^{(N)}_{\mathrm{inv}}$:

$$\hat{\mathbf{J}}^{(N)}_{\mathrm{inv}} \triangleq \frac{1}{N}\sum_{j=1}^{N}\left[\mathrm{E}_{Z|XY}\left[\nabla_x \log p(\hat{x}^{(j)}, Z, \hat{y}^{(j)})\right] \cdot \mathrm{E}_{Z|XY}\left[\nabla_x \log p(\hat{x}^{(j)}, Z, \hat{y}^{(j)})\right]^T\right]^{-1}. \tag{5.393}$$

The bound (5.390)–(5.392) is eventually evaluated as:

$$\begin{aligned}\mathbf{E} &\triangleq \mathrm{E}_{XY}[(\hat{x}(y)-X)(\hat{x}(y)-X)^T]\\ &\succeq \hat{\mathbf{J}}^{(N)}_{\mathrm{inv}}.\end{aligned} \tag{5.394}$$

In words: the bound (5.394) is determined by *averaging inverse* matrices, whereas the bound (5.382) is determined as the *inverse of an average* matrix. The bound (5.394) involves numerous matrix inversions, whereas the bound (5.382) requires only *one* matrix inversion; the bound (5.394) is therefore more costly to evaluate (but tighter!) than (5.382).

5.4.5 Hybrid BCRB

We now consider a system with random vectors X and Z, and parameters Θ; the joint pdf is given by $p(x,z,y|\theta)$. We wish to compute the hybrid BCRB for X:

$$\mathrm{E}_{XY|\Theta}[(\hat{x}(Y)-X)(\hat{x}(Y)-X)^T] \triangleq \mathbf{E}^{(X\Theta)}_{22} \succeq \left[\mathbf{H}^{-1}(\theta)\right]_{22}, \tag{5.395}$$

where

$$\mathbf{H}(\theta) \quad \stackrel{\triangle}{=} \quad \begin{bmatrix} \mathbf{H}_{11}(\theta) & \mathbf{H}_{12}(\theta) \\ \mathbf{H}_{21}(\theta) & \mathbf{H}_{22}(\theta) \end{bmatrix}, \tag{5.396}$$

$$\mathbf{H}_{11}(\theta) = \mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta) \nabla_\theta^T \log p(X,Y|\theta)\right] \tag{5.397}$$
$$\mathbf{H}_{12}(\theta) = \mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta) \nabla_x^T \log p(X,Y|\theta)\right] \tag{5.398}$$
$$\mathbf{H}_{12}(\theta) = [\mathbf{H}_{21}(\theta)]^T \tag{5.399}$$
$$\mathbf{H}_{22}(\theta) = \mathrm{E}_{XY|\Theta}\left[\nabla_x \log p(X,Y|\theta) \nabla_x^T \log p(X,Y|\theta)\right], \tag{5.400}$$

and

$$p(x,y|\theta) \stackrel{\triangle}{=} \int_z p(x,z,y|\theta)dz. \tag{5.401}$$

The expressions (5.397)–(5.400) can be evaluated as follows.

Lemma 5.6. If the integral $\int_z p(x,z,y|\theta)dz$ is differentiable under the integral sign (w.r.t. x and θ), then

$$\mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta) \nabla_\theta^T \log p(X,Y|\theta)\right]$$
$$= \mathrm{E}_{XY|\Theta}\Big[\mathrm{E}_{Z|XY\Theta}[\nabla_\theta \log p(X,Z,Y|\theta)]\, \mathrm{E}_{Z|XY\Theta}[\nabla_\theta \log p(X,Z,Y|\theta)]^T\Big] \tag{5.402}$$

$$\mathrm{E}_{XY|\Theta}\left[\nabla_\theta \log p(X,Y|\theta) \nabla_x^T \log p(X,Y|\theta)\right]$$
$$= \mathrm{E}_{XY|\Theta}\Big[\mathrm{E}_{Z|XY\Theta}[\nabla_\theta \log p(X,Z,Y|\theta)]\, \mathrm{E}_{Z|XY\Theta}[\nabla_x \log p(X,Z,Y|\theta)]^T\Big] \tag{5.403}$$

$$\mathrm{E}_{XY|\Theta}\left[\nabla_x \log p(X,Y|\theta) \nabla_x^T \log p(X,Y|\theta)\right]$$
$$= \mathrm{E}_{XY|\Theta}\Big[\mathrm{E}_{Z|XY\Theta}[\nabla_x \log p(X,Z,Y|\theta)]\, \mathrm{E}_{Z|XY\Theta}[\nabla_x \log p(X,Z,Y|\theta)]^T\Big]. \tag{5.404}$$

□

A numerical algorithm to compute the bound (5.395) may readily be extracted from Lemma 5.6. We leave the details as an exercise for the reader.

5.4.6 Parameter Estimation in State-Space Models

In Section 5.3.7, we determined BCRBs for estimation in state-space models. We derived those bounds from the joint pdf $p(x, \theta, y)$. Here, we derive such bounds from marginal pdfs. We investigate the standard unconditional BCRB for Θ (cf. (5.377)):

$$\begin{aligned}
&\mathrm{E}_{\Theta Y}[(\hat{\theta}(Y) - \Theta)(\hat{\theta}(Y) - \Theta)^T] \\
&\qquad \succeq \mathbf{J}^{-1} && (5.405)\\
&\qquad \triangleq \mathrm{E}_{\Theta Y}\left[\nabla_\theta \log p(\Theta, Y) \nabla_\theta^T \log p(\Theta, Y)\right]^{-1} && (5.406)\\
&\qquad = -\mathrm{E}_{\Theta Y}\big[\nabla_\theta \nabla_\theta^T \log p(\Theta, Y)\big]^{-1}, && (5.407)
\end{aligned}$$

where

$$p(\theta, y) \triangleq \int_x p(\theta, x, y) dx. \tag{5.408}$$

First, we consider the state-space model with constant parameters; its pdf $p(\theta, x, y)$ equals (see Fig. 5.18(a)):

$$p(\theta, x, y) \triangleq p_\Theta(\theta) p_0(x_0) \prod_{k=1}^{N} p(x_k | x_{k-1}, \theta) p(y_k | x_k). \tag{5.409}$$

By applying Lemma 5.4, we have:

$$\begin{aligned}
&\mathrm{E}_{\Theta Y}\big[\nabla_\theta \log p(\Theta, Y) \nabla_\theta^T \log p(\Theta, Y)\big] \\
&\qquad = \mathrm{E}_{\Theta Y}\Big[\mathrm{E}_{X|\Theta Y}[\nabla_\theta \log p(\Theta, X, Y)]\, \mathrm{E}_{X|\Theta Y}[\nabla_\theta \log p(\Theta, X, Y)]^T\Big].
\end{aligned} \tag{5.410}$$

We rewrite the expression $\mathrm{E}_{X|\Theta Y}[\nabla_\theta \log p(\Theta, X, Y)]$ in the RHS of (5.410) as:

$$\begin{aligned}
&\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(\Theta, X, Y)\right] \\
&\qquad = \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p_\Theta(\theta)\right] + \sum_{k=1}^{N} \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X_k | X_{k-1}, \theta)\right] && (5.411)\\
&\qquad = \nabla_\theta \log p_\Theta(\theta) + \sum_{k=1}^{N} \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X_k | X_{k-1}, \theta)\right]. && (5.412)
\end{aligned}$$

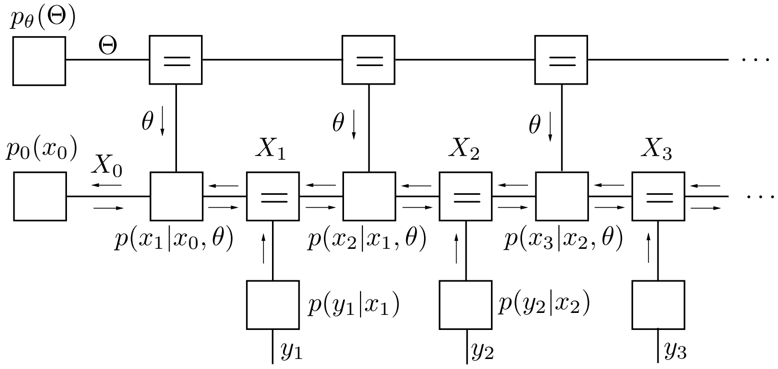


Figure 5.25: Unconditional BCRB for estimating (constant) parameters of a state-space model.

The expression $\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X_k|X_{k-1},\theta)\right]$ in the RHS of (5.412) can be computed as:

$$\begin{aligned}
&\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X_k|X_{k-1},\theta)\right]\\
&\quad= \mathrm{E}_{X_{k-1}X_k|\Theta Y}\left[\nabla_\theta \log p(X_k|X_{k-1},\theta)\right] && (5.413)\\
&\quad= \int_{x_{k-1}}\int_{x_k} p(x_k,x_{k-1}|\theta,y)\nabla_\theta \log p(x_k|x_{k-1},\theta)dx_{k-1}dx_k, && (5.414)
\end{aligned}$$

where the joint pdf $p(x_k,x_{k-1}|\theta,y)$ may be determined by the sum-product algorithm, as depicted in Fig. 5.25:

$$p(x_k,x_{k-1}|\theta,y) = \frac{\mu_{X_{k-1}\to p_k}(x_{k-1})\mu_{X_k\to p_k}(x_k)p(x_k|x_{k-1},\theta)}{\int_{x_{k-1}}\int_{x_k}\mu_{X_{k-1}\to p_k}(x_{k-1})\mu_{X_k\to p_k}(x_k)p(x_k|x_{k-1},\theta)dx_{k-1}dx_k}. \tag{5.415}$$

As a consequence, the expression (5.410) may be evaluated by the following algorithm:

a) Generate a list of samples $\{\hat{\theta}^{(j)},\hat{y}^{(j)}\}_{j=1}^N$ from $p(\theta,y)$.

b) For $j=1,\ldots,N$:

i) Compute the messages $\mu_{X_k \to p_{k-1}}$ and $\mu_{X_k \to p_k}$ by a forward and backward sum-product sweep.

ii) Evaluate the expression:

$$\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X, \hat{\theta}^{(j)}, \hat{y}^{(j)})\right], \tag{5.416}$$

using (5.411)–(5.415), and the messages computed in i).

c) Compute the matrix $\hat{\mathbf{J}}^{(N)}$:

$$\hat{\mathbf{J}}^{(N)} \triangleq \frac{1}{N} \sum_{j=1}^{N} \left[\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X, \hat{\theta}^{(j)}, \hat{y}^{(j)})\right] \cdot \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(X, \hat{\theta}^{(j)}, \hat{y}^{(j)})\right]^T \right]. \tag{5.417}$$

Eventually, we evaluate the bound (5.405)–(5.407) as:

$$\mathrm{E}_{\Theta Y}[(\hat{\theta}(Y) - \Theta)(\hat{\theta}(Y) - \Theta)^T] \succeq \left[\hat{\mathbf{J}}^{(N)}\right]^{-1}. \tag{5.418}$$

The algorithm can straightforwardly be extended to state-space models with *time-dependent* parameters: one needs to slightly modify the expressions (5.412)–(5.415); the computation of (5.416) then involves a forward and backward sum-product sweep in the factor graph of Fig. 5.26.

Example 5.10. (CRBs for estimation in AR model)
In Example 5.9, we computed hybrid CRBs for the joint estimation of the state and parameters of AR models; we derived those hybrid CRBs from the information matrix of the *joint* pdf of the AR model. Here, we derive Cramér-Rao-type bounds (for the same estimation problem) from the information matrix of *marginals*. We outline a (numerical) method to compute the CRB (5.368) for the parameters $a_1, \ldots, a_M$, σ_U^2 and σ_W^2. In Appendix K, we propose a similar method to compute the hybrid CRB (5.395) for X_k. We compute both bounds *numerically*, since they are not tractable analytically; on the other hand, the bounds of Example 5.9 are tractable, but unfortunately loose.

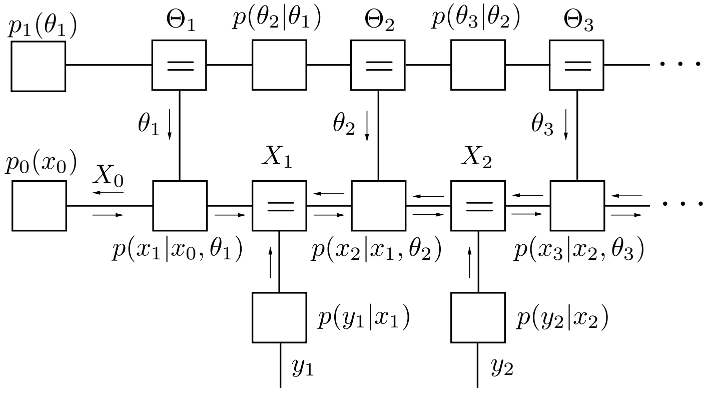


Figure 5.26: Unconditional BCRB for estimating (time-dependent) parameters of a state-space model.

The CRB (5.368) for $\Theta \stackrel{\triangle}{=} (a_1, \ldots, a_M, \sigma_U^2, \sigma_W^2)$ can be determined by the algorithm of Section 5.4.1. We outline how the expression (5.371) may be evaluated for the AR model (5.343). Substituting (5.343) in (5.371) amounts to:

$$\begin{aligned}\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log p(\theta, X, y)\right] = &\sum_{k=1}^{N} \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_1(X_k, \ldots, X_{k-M}, \mathbf{a}, \sigma_U^2)\right] \\ &+ \sum_{k=1}^{N} \mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_2(X_k, y_k, \sigma_W^2)\right], \end{aligned} \tag{5.419}$$

where f_1 and f_2 are given by (5.350) and (5.354) respectively. The expression $\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_{1,k}\right]$ and $\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_{2,k}\right]$ in the RHS of (5.419) can be computed from sum-product messages arriving at the nodes f_1 and f_2 (cf. (5.414) and (5.415)). Those messages are computed for a given observation vector y, and for *given* values of the parameters $\mathbf{a}, \sigma_W^2$, and σ_U^2.[10] As a consequence, the sum-product messages are available in closed-form: they are Gaussian messages, computed in a forward and backward Kalman recursion (cf. Appendix H). In Appendix K, we explain how the expectations $\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_{1,k}\right]$ and $\mathrm{E}_{X|\Theta Y}\left[\nabla_\theta \log f_{2,k}\right]$

[10]In the estimation problem at hand, the parameters are obviously *not* given; they need to be estimated.

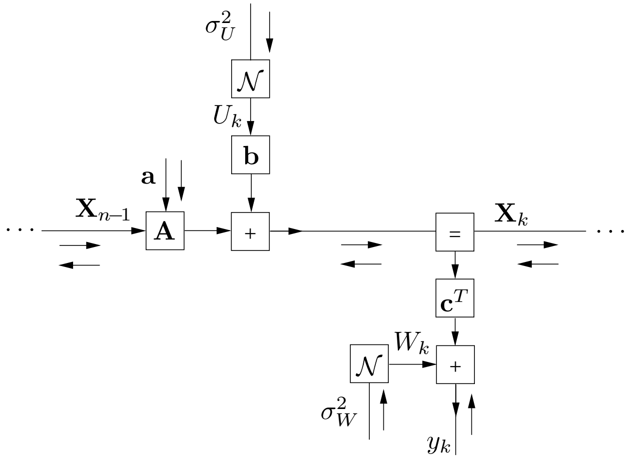


Figure 5.27: Computing Gaussian messages (along the edges X_k) in a forward and backward Kalman recursion, for a given observation vector y, and given parameters $\mathbf{a}$, σ_W^2, and σ_U^2.

may be computed from the Gaussian messages.

In summary, the algorithm we propose for computing the bound (5.368) for $\Theta \triangleq (a_1, \ldots, a_M, \sigma_U^2, \sigma_W^2)$ performs the following steps:

a) Generate a list of samples $\{\hat{y}^{(j)}\}_{j=1}^N$ from $p(y|\mathbf{a}, \sigma_W^2, \sigma_U^2)$.

b) For $j = 1, \ldots, N$:

   i) Compute the Gaussian messages $\mu_{X_k \to f_{1,k}}$ and $\mu_{X_k \to f_{2,k}}$ by a forward and backward Kalman recursion.

   ii) Evaluate the expression:

$$\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2 \sigma_U^2\, Y}\Big[\nabla_\theta \log p(X, \hat{y}^{(j)}|\mathbf{a}, \sigma_W^2, \sigma_U^2)\Big] \tag{5.420}$$

   using (K.152)–(K.157), and the messages computed in bi.

c) Compute the matrix $\hat{\mathbf{F}}^{(N)}(\theta)$:

$$\hat{\mathbf{F}}^{(N)}(\theta) \triangleq \frac{1}{N} \sum_{j=1}^{N} \Bigg[\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2 \sigma_U^2\, Y}\Big[\nabla_\theta \log p(X, \hat{y}^{(j)}|\mathbf{a}, \sigma_W^2, \sigma_U^2)\Big]$$

$$\cdot\, \mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[\nabla_\theta \log p(X, \hat{y}^{(j)}|\mathbf{a}, \sigma_W^2, \sigma_U^2)\Big]^T\Bigg]. \tag{5.421}$$

The CRB (5.368) for Θ is computed as the inverse of $\hat{\mathbf{F}}^{(N)}(\theta)$.

Along similar lines, one can compute the hybrid CRB (5.395) for X_k ($k = 1, \ldots, N$) from the marginal $p(x_k, y|\mathbf{a}, \sigma_W^2, \sigma_U^2)$. We refer to Appendix K for more information.

In Fig. 5.28 to Fig. 5.35, we present numerical results. In Fig. 5.28, the standard CRB for a is shown for several values of σ_W^2 and N; we consider the case where σ_U^2 and σ_W^2 are *known* as well as the case where those two parameters are *unknown*, i.e., where they need to be *estimated.* In Fig. 5.29, we compare the standard CRB for a (for the case where σ_U^2 and σ_W^2 are unknown) to the MSE of a practical estimation algorithm, i.e., the grid-based algorithm of [101] that *jointly* estimates the state X and the parameters a, σ_U^2 and σ_W^2. The grid-based algorithm of [101] is complex, therefore, we could only average its squared estimation error over 100 simulations; the dashed curves are therefore rather "noisy".

In Fig. 5.30, the standard CRB and the hybrid CRB for σ_U^2 (with unknown a and σ_W^2) are shown for several values of σ_W^2 and N. As we pointed out before, the hybrid CRB for σ_U^2 does not depend on σ_W^2. In Fig. 5.31, we show the standard CRB for σ_U^2 for the case of known (unknown) a and σ_W^2. In Fig. 5.32, we compare the standard CRB for σ_U^2 (with unknown a and σ_W^2) to the MSE of the grid-based algorithm of [101].

The standard CRB and the hybrid CRB for σ_W^2 are shown in Fig. 5.33 for several values of σ_W^2 and N (where a and σ_U^2 are assumed to be unknown). In Fig. 5.31, the standard CRB for σ_W^2 for the case of known/unknown a and σ_U^2 are shown. Fig. 5.35 shows the standard CRB for σ_W^2 (with unknown a and σ_U^2) together with the MSE of the grid-based algorithm of [101].

Note that the standard CRBs are nicely tight; they allow us to certify that the grid-based algorithm of [101] is close to optimal.

$\square$

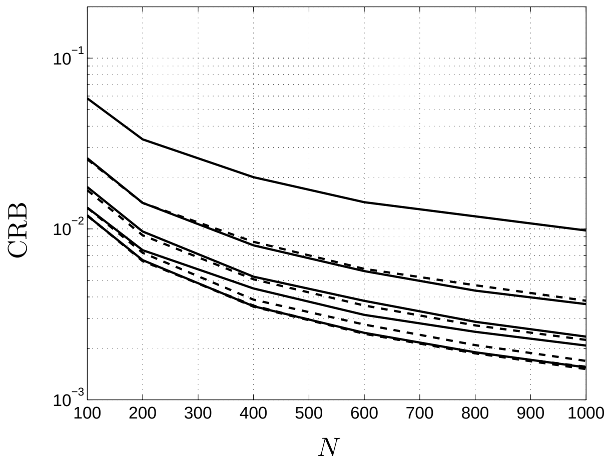


Figure 5.28: Standard CRB for a with known (dashed) and unknown (solid) σ_U^2 and σ_W^2; $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001/0.0001/0.00001$. The CRBs for *known* $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.0001, 0.00001$ coincide (dashed lines).

5.4.7 Code-Aided Channel Estimation

We now consider the problem of code-aided channel estimation; symbols U_k, protected by an error-correcting code (with indicator function $I(u)$), are transmitted over a channel with memory; the state of the channel is denoted by X_k, whereas Y_k stands for the channel observation at time k. The probability function of this system is given by

$$p(u, x, y) = p_0(x_0) \prod_{k=1}^{N} p(x_k, y_k | u_k, x_{k-1}) I(u), \tag{5.422}$$

as depicted in Fig. 5.36. The box at the top represents the indicator function $I(u)$, the row of nodes at the bottom stands for the factors $p_0(x_0)$ and $p(x_k, y_k|u_k, x_{k-1})$. We wish to compute the standard unconditional BCRB for the estimation of X_k (the extension to other BCRBs is straightforward).

In principle, one could compute the standard unconditional y the algorithm of Section 5.4.3 (with $Z = U$). With the pdf (5.422), the expec-

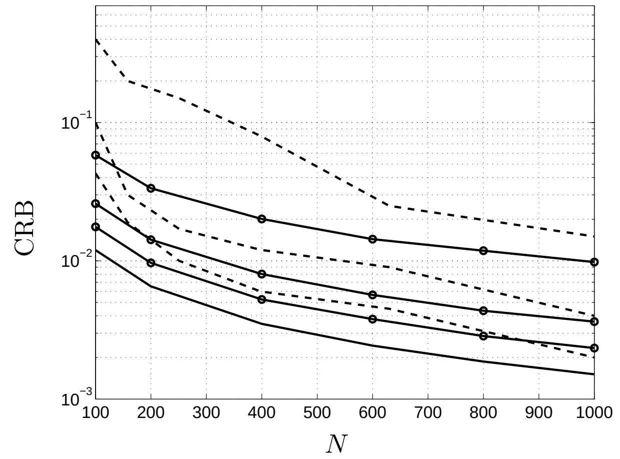


Figure 5.29: Standard CRB for a with unknown σ_U^2 and σ_W^2 (solid) together with the MSE of the grid-based algorithm of [100] (dashed); $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001$. Also shown is the standard CRB for a with *known* $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0$ (dashed-dotted line).

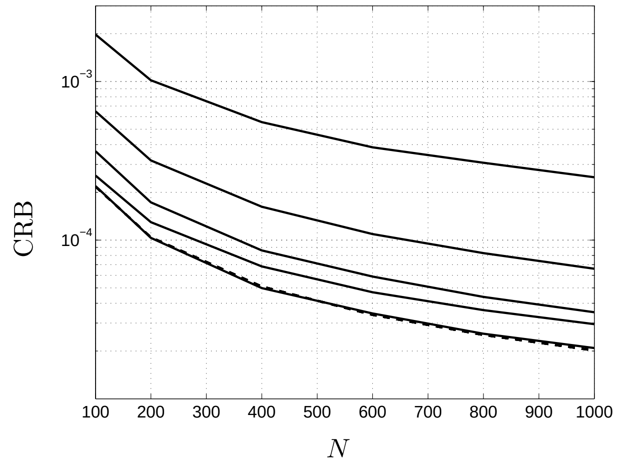


Figure 5.30: Hybrid CRB (dashed) and standard CRB (solid) for σ_U^2; $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001/0.0001/0.00001$.

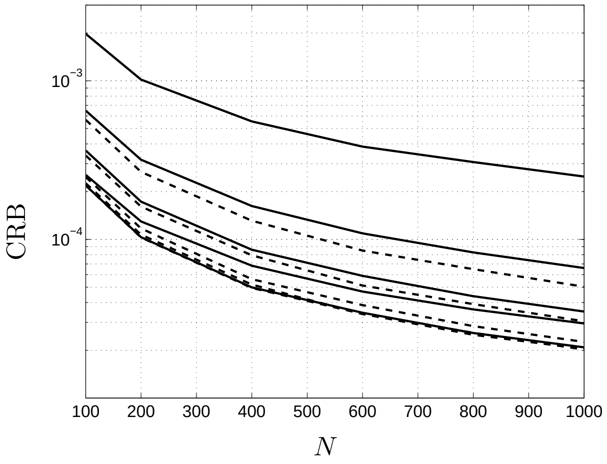


Figure 5.31: Standard CRB for σ_U^2; $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001/0.0001/0.00001$; known (dashed) and unknown (solid) σ_W^2 and a.

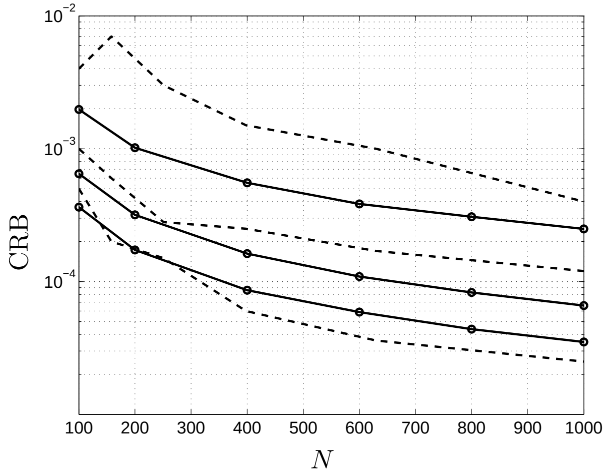


Figure 5.32: Standard CRB (solid) for σ_U^2 together with the MSE of the grid-based algorithm of [100] (dashed); $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001$.

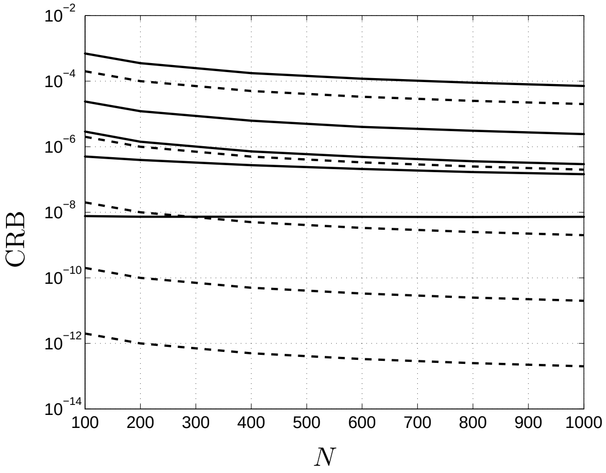


Figure 5.33: Hybrid CRB (dashed) and standard CRB (solid) for σ_W^2; $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001/0.0001/0.00001$.

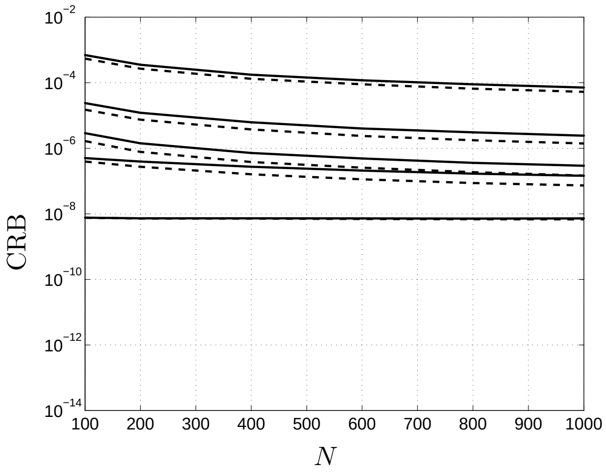


Figure 5.34: Standard CRB for σ_W^2; $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001/0.0001/0.00001$; known (dashed) and unknown (solid) σ_W^2 and a.

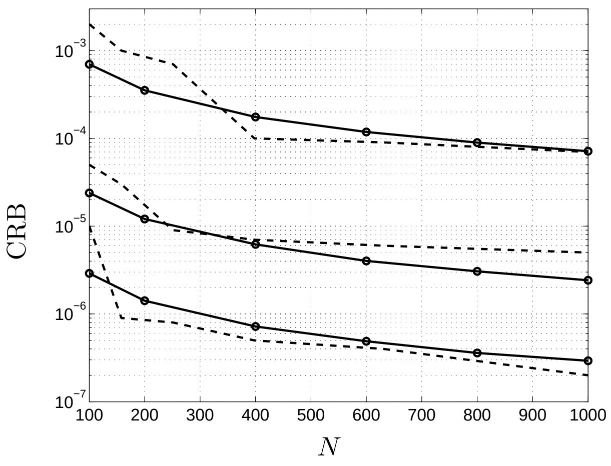


Figure 5.35: Standard CRB (solid) for σ_W^2 together with the MSE of the grid-based algorithm of [100] (dashed); $\sigma_U^2 = 0.1$ and $\sigma_W^2 = 0.1/0.01/0.001$.

tation $\mathrm{E}_{U|XY}[\nabla_x \log p(X, U, Y)]$ in the RHS of (5.379) can be evaluated as:

$$\mathrm{E}_{U|XY}[\nabla_x \log p(X, U, Y)]$$
$$= \nabla_x \log p_0(X_0) + \sum_{k=1}^{N} \mathrm{E}_{U_k|XY}[\nabla_x \log p(X_k, Y_k|X_{k-1}, U_k)] \,. \quad (5.423)$$

An expression similar to (5.423) has been derived in [148]. The second term in the RHS of (5.423) involves the marginals $p(u_k|x, y)$. For most practical probabilistic codes (e.g., LDPC codes or turbo-codes), those marginals are hard to obtain—with exception of (short-memory) convolutional codes. Note that replacing the exact marginals by marginals by *approximations* obtained from an iterative decoder (as suggested in, e.g., [148] for standard CRBs) does not amount to the (exact) BCRBs. One may obtain accurate approximations of the marginals by means of Gibbs sampling, which is, however, a time-consuming procedure.

Whereas it is often difficult to compute the exact BCRBs for code-aided channel estimation, it is usually feasible to derive upper and lower bounds on the BCRB. Upper bounds are obtained by assuming that the symbols are independent and *uniformly* distributed (i.u.d.). On the other hand,

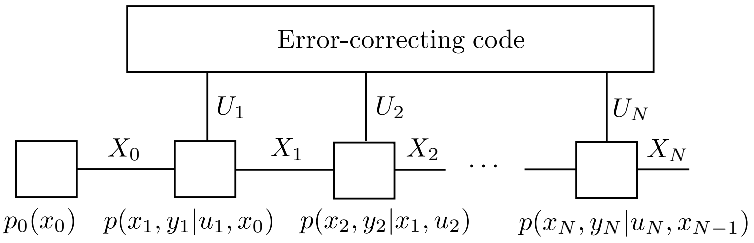


Figure 5.36: Code-aided channel estimation.

if one assumes that the symbols are *known*, one obtains lower bounds (so-called "modified" BCRBs (MBCRB)). At sufficiently high SNR, the modified BCRBs coincide with the (true) BCRBs; at high SNR, the bit (and frame) error rates are low (e.g., BER $< 10^{-2}$), and the symbols can therefore be considered as (almost) "known". Communications systems usually operate at low bit (and frame) error rates, therefore, *modified* BCRBs typically suffice.

Example 5.11. (Modulated random-walk phase model)
We consider here the (modulated) random-walk phase model (4.1)–(4.2). The BCRBs for this model are intractable, however, modified BCRBs are easy to obtain. In fact, we already computed the modified (standard unconditional) BCRB in Example 5.6, where we investigated the *unmodulated* random-walk phase model (cf. Fig. 5.13). In Fig. 5.37, we show this modified BCRB (for the MSE averaged over a block of length 100) together with:

- an upper bound based on i.u.d. symbols,
- a result obtained by substituting approximate marginals (from an LDPC-decoder) in (5.423),
- the MSE of a practical message-passing estimator, i.e., an estimator that uses adaptive quantization (see Section 4.7.2).

As can be seen from Fig. 5.37, the upper bound (based on i.u.d. symbols) is loose; the curve obtained from approximative marginals is (at low SNR) *not* a lower bound on the BCRB: for SNR values below -1dB, the curve

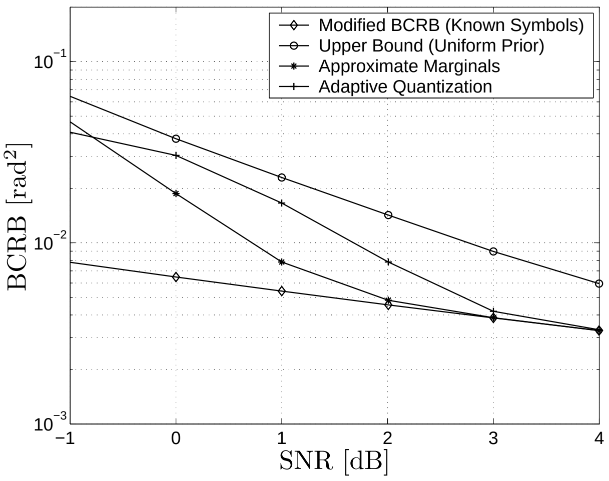


Figure 5.37: Lower and upper bounds on the BCRB for the modulated random-walk phase model with $N = 100$, and $\sigma_W^2 = 10^{-4}$ rad^2.

lies *above* the MSE of the practical estimator. At high SNR (i.e., SNR $\geq$ 3dB), the curve obtained from approximative marginals coincides with the modified BCRB. Also the MSE of the practical estimator coincides with the modified BCRB for those SNR values. In other words, we can certify that the practical estimator is close to optimal for SNR $\geq$ 3dB. The latter SNR region is the most relevant: the BER at 3dB and 4dB is about 10^{-2} and 10^{-3} respectively (not shown here); at SNR values below 3dB, the error rates are too high for most practical purposes.

In Fig. 5.38, we show the modified BCRB for filtering and smoothing as a function of the position k inside the block (for $\sigma_N = 0.446$ rad (4dB) and $\sigma_W = 10^{-2}$ rad); also shown is the MSE of the practical phase estimator. It can be seen from this figure that the MSE of the practical estimator coincides with the modified BCRB . $\square$

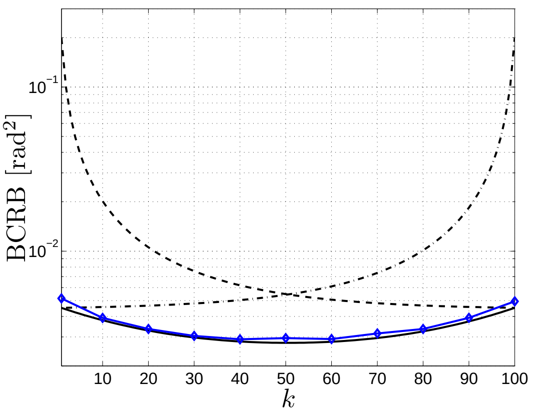


Figure 5.38: Modified BCRBs for the modulated random-walk phase model with $N = 100$, $\sigma_N = 0.446$ rad (4dB) and $\sigma_W = 10^{-2}$ rad; Shown are the MBCRB of the forward sweep, i.e., filtering (dashed line), MBCRB of the backward sweep (dashed-dotted), and MBCRB of smoothing (solid). Also shown is the MSE of the phase estimator that uses adaptive quantization (diamonds).

5.5 Summary

We summarize here the main results of this chapter.

- We started by reviewing the **three main types** of Cramér-Rao bounds (CRB): **standard**, **Bayesian**, and **hybrid** Cramér-Rao bounds, which are applicable to parameters, random variables and the joint estimation of parameters and random variables respectively.
- We outlined the **two main strategies** to compute Cramér-Rao bounds: one may derive Cramér-Rao bounds from the inverse information matrix of the **joint** pdf; alternatively, one may obtain such bounds from inverse information matrices of **marginals**.
- We proposed **novel algorithms** to compute CRBs:
  - following each of both strategies,
  - for each of the three types of CRBs.

  Our methods are **message-passing** algorithms, operating on the factor graph of the system at hand. We derived the **local update rules**, each time starting from a simple working example.
- We treated several **applications**:
  - filtering in (general) state-space models,
  - smoothing in (general) state-space models,
  - estimation of the parameters of state-space models,
  - code-aided channel estimation.

  We considered various **examples** of state-space models:
  - linear and non-linear dynamial systems,
  - the (modulated and unmodulated) random-walk phase model,
  - AR models.

5.6 Outlook

- In this chapter, we have exclusively focused on the squared error loss function. Cramér-Rao-type bounds have recently been derived for other types of **loss functions** , i.e., for loss functions that are bounded below by a function of the form $g(x) = |x|^\ell$ for $\ell \geq 2$ [167]. It needs be verified whether our methods can also be applied to Cramér-Rao-type bounds for those loss functions.

- Our methods seem to apply to several other **types of bounds**, i.e., Weiss-Weinstein bounds [208], Bobrobsky-Zakai bounds [25], and Bhattacharyya bounds [21]. Those bounds are often tighter (at low SNR) than the Cramér-Rao-type bounds, however, they are more complicated. Recently, Reece et al. evaluated those three bounds for the problem of filtering in state-space models [169] (see also [168]); they obtained forward recursions that are similar to the forward recursion of [192], i.e., the standard unconditional BCRB for filtering. It should be straightforward to extend the results of [169] (and [168]) to general estimation problems, following the line of thought of this chapter. Interestingly, the Weiss-Weinstein bound also applies to **discrete variables**.

- Information matrices are not only the key to Cramér-Rao-type bounds, they also play an important role in **information geometry**. Information geometry deals a.o. with the geometry of the (non-Euclidean) space of parametrized statistical models (with parameters Θ). The (unique) invariant metric of this parameter space turns out to be the **Fisher information matrix**. By means of the Fisher information matrix, one can for example define the *distance* between distributions or the *steepest-descent direction* (in the parameter space) of a function $f(\theta)$. Since the parameter space is non-Euclidian ("curved"), the steepest-descent direction of $f(\theta)$ is not given by its (Euclidean) gradient, but by its **natural gradient** [7] [8], defined as

$$\tilde{\nabla}_\theta f(\theta) \stackrel{\triangle}{=} \mathbf{F}^{-1}(\theta)\nabla_\theta f(\theta). \tag{5.424}$$

  Amari [7] proposed optimization algorithms based on the natural gradient ("natural-gradient adaption"); for several interesting applications (e.g., training of feed-forward neural networks), those optimization methods converge significantly faster (up to three orders of magnitude!) than the standard gradient methods (based on the Euclidean gradient) [156]. The chief drawback of natural-gradient adaption, however, is that it requires the inverse of the Fisher information matrix. For many sophisticated probabilistic models, this matrix is not tractable. However, as we have shown, the Fisher information matrix can often readily be computed numerically, which could lead to novel natural-gradient algorithms.

Note that natural-gradient algorithms may conveniently be derived in context of **factor graphs** and **summary propagation**; we have shown in Section 4.8 how (Euclidean) gradients can be computed by message passing; in this chapter, we have developed message-passing algorithms for computing Fisher information matrices (cf. Section 5.4).

As we pointed out, the computation of Fisher information matrices often involves sum-product messages. If it is not feasible to compute the latter exactly, they may be computed approximately by sum-product message passing on **cyclic graphs**; this amounts to **approximate Fisher information matrices**, which, obviously, do not lead to the correct Cramér-Rao bounds; those approximate Fisher information matrices, however, can be the basis for (low-complexity) natural-gradient algorithms.

A potential application is estimation in **single- or multi-channel AR(MA) models** (cf. Example 5.10), which are useful models for **blind source separation** and **blind deconvolution** [34]; recursive ("online") natural-gradient algorithms for (single- or multi-channel) AR(MA) models can readily be derived (cf. Example 5.10); simulations are required to assess the complexity-performance trade-off of such algorithms. More generally, natural-gradient algorithms may become an (even more) important tool in model-based signal processing.

- The Fisher information matrix can also be used to derive so-called **Fisher kernels** (proposed by Jaakkola et al. [89]) from probabilistic models.[11] For a given probability model $p(y|\theta)$ with parameters Θ and observations Y, the Fisher kernel is defined as:

$$\kappa(y_i, y_j) \overset{\triangle}{=} \nabla_\theta^T \log p(y_i|\theta)\mathbf{F}^{-1}(\theta)\nabla_\theta \log p(y_j|\theta). \tag{5.425}$$

The main difficulty is also here the computation of the inverse information matrix. Our methods for computing information matrix

[11]For a brief introduction to kernel methods and for a discussion on how kernels can be computed from graphical models, we refer to Appendix D.

allow us to derive Fisher kernels from sophisticated probabilistic models—also for this purpose, one may use iterative sum-product message passing, amounting to *approximate* Fisher information matrices. The inversion of (dense) Fisher information matrices can be carried out approximately but *efficiently* by imposing **structure** on those matrices.

In general, our methods for computing Fisher information matrices (with our without approximations) may lead to novel kernel-based algorithms for classification, compression and clustering of images and biomedical and speech signals.

Chapter 6

Computing Information Rates of Continuous Channels with Memory

We present here a numerical method to compute information rates of continuous channels with memory. We will apply the method to the random-walk phase model. The results of this chapter are based on [49].

6.1 Introduction

We consider the problem of computing the information rate[1]

$$I(X;Y) \triangleq \lim_{n\to\infty} \frac{1}{n} I(X_1,\ldots,X_n;Y_1,\ldots,Y_n) \tag{6.1}$$

between the input process $X = (X_1, X_2, \ldots)$ and the output process $Y = (Y_1, Y_2, \ldots)$ of a time-invariant discrete-time channel with memory. Let $x_k^n \triangleq (x_k, x_{k+1}, \ldots, x_n)$ and $x^n \triangleq (x_1, x_2, \ldots, x_n)$. We will assume

[1]Basic notions from information theory are reviewed in Appendix B.

that there is an ergodic stochastic process $S = (S_0, S_1, S_2, \ldots)$ such that

$$p(x^n, y^n, s_0^n) = p(s_0) \prod_{k=1}^{n} p(x_k, y_k, s_k | s_{k-1}) \tag{6.2}$$

for all $n > 0$ and with $p(x_k, y_k, s_k|s_{k-1})$ not depending on k.

For *finite* input alphabet $\mathcal{X}$ (= range of X_k) and *finite* state space $\mathcal{S}$ (= range of S_k), a practical method for computing the information rate (6.1) was proposed in [12] [179] and [160].[2] In [13], this method was described in greater generality and extended to the computation of upper and lower bounds on the information rate of very general channels (see also [10]). Another method to compute an upper bound was presented in [11]. A method to *approximately* compute information rates of finite-state channels is proposed in [181]; and it is based on generalized belief propagation [223].

In this chapter, we extend the methods of [12] and [13] to continuous state spaces $\mathcal{S}$. For the sake of clarity, we will assume that $\mathcal{S}$ is a bounded subset of $\mathbb{R}^\nu$, the ν-dimensional Euclidean space; the input alphabet $\mathcal{X}$ may also be continuous. The key to this extension is the use of Monte-Carlo integration methods ("particle filters") [59] (cf. Section 4.6.4).

This chapter is structured as follows. In Section 6.2, we review the basic idea of [12] as presented in [13]. In Section 6.3, we show how particle methods allow to deal with a continuous state space; in Section 6.4, we apply the resulting algorithm to the random-walk phase model. In Section 6.5, we summarize this chapter.

6.2 Review of Basic Method

We briefly review the basic idea of [12] as presented in [13]. We first note that, as a consequence of the Shannon-McMillan-Breiman theorem (cf. Theorem B.6 and B.7), the sequence $-\frac{1}{n}\log p(X^n)$ converges with probability 1 to the entropy rate $H(X)$, the sequence $-\frac{1}{n}\log p(Y^n)$ converges with probability 1 to the differential entropy rate $h(Y)$, and the sequence $-\frac{1}{n}\log p(X^n, Y^n)$ converges with probability 1 to $H(X) +$

[2]See [62] for an alternative approach).

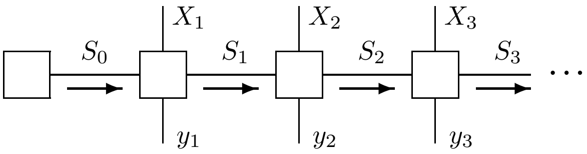


Figure 6.1: Computation of $p(y^n)$ by message passing through the factor graph of (6.2).

$h(Y|X)$. From these observations, the quantity $I(X;Y) = h(Y) - h(Y|X)$ can be computed as follows:

a) Sample two “very long” sequences x^n and y^n.

b) Compute $\log p(x^n)$, $\log p(y^n)$, and $\log p(x^n, y^n)$. If $h(Y|X)$ is known analytically, then it suffices to compute $\log p(y^n)$.

c) Conclude with the estimate

$$\hat{I}(X;Y) \triangleq \frac{1}{n}\log p(x^n, y^n) - \frac{1}{n}\log p(x^n) - \frac{1}{n}\log p(y^n) \tag{6.3}$$

or, if $h(Y|X)$ is known analytically, $\hat{I}(X;Y) \triangleq -\frac{1}{n}\log p(y^n) - h(Y|X)$.

The computations in Step 2 can be carried out by forward sum-product message passing through the factor graph of (6.2), as is illustrated in Fig. 6.1. If the state space $\mathcal{S}$ is finite, this computation is just the forward sum-product recursion of the BCJR algorithm [15].

Consider, for example, the computation of

$$p(y^n) = \int_{x^n}\int_{s_0^n} p(x^n, y^n, s_0^n), \tag{6.4}$$

where $\int_x g(x)dx$ stands for the summation of $g(x)$ over its support if x is discrete, otherwise, it stands for integration. Define the state metric $\mu_k(s_k) \triangleq p(s_k, y^k)$. By straightforward application of the sum-product algorithm, we recursively compute the messages (state metrics)

$$\begin{aligned}\mu_k(s_k) &= \int_{x_k}\int_{s_{k-1}} \mu_{k-1}(s_{k-1})\, p(x_k, y_k, s_k|s_{k-1})dx_k ds_{k-1} & (6.5)\\ &= \int_{x^k}\int_{s_0^{k-1}} p(x^k, y^k, s^k)dx^k ds_0^{k-1} & (6.6)\end{aligned}$$

for $k = 1, 2, 3, \ldots$ The desired quantity (6.4) is then obtained as

$$p(y^n) = \int_{s_n} \mu_n(s_n), \tag{6.7}$$

the sum of (or the integral over) all final state metrics.

For large k, the state metrics μ_k computed according to (6.5) quickly tend to zero. In practice, the recursion (6.5) is therefore changed to

$$\mu_k(s_k) = \lambda_k \int_{x_k}\int_{s_{k-1}} \mu_{k-1}(s_{k-1})\, p(x_k, y_k, s_k|s_{k-1})dx_k ds_{k-1}, \tag{6.8}$$

where λ_1, λ_2, ...are positive scale factors (cf. Remark3.3). We will choose these factors such that

$$\int_{s_k} \mu_k(s_k) = 1 \tag{6.9}$$

holds for all k. It then follows that

$$\frac{1}{n}\sum_{k=1}^{n} \log \lambda_k = -\frac{1}{n}\log p(y^n). \tag{6.10}$$

The quantity $-\frac{1}{n}\log p(y^n)$ thus appears as the average of the logarithms of the scale factors, which converges (almost surely) to $h(Y)$.

If necessary, the quantities $\log p(x^n)$ and $\log p(x^n, y^n)$ can be computed by the same method, see [13].

6.3 A Particle Method

If both the input alphabet $\mathcal{X}$ and the state space $\mathcal{S}$ are finite sets, then the method of the previous section is a practical algorithm. However, we are now interested in the case where $\mathcal{S}$ (and perhaps also $\mathcal{X}$) are continuous, as stated in the introduction. In this case, the computation of (6.8) is a problem.

This problem can be addressed by Monte-Carlo methods known as particle filtering [59]. Such algorithms may be viewed as message-passing algorithms where the messages (which represent probability distributions) are represented by a list of samples ("particles") from the distribution (cf. Section 4.6.4). In particular, we will represent the message μ_k by a list $\{\hat{s}_{k,\ell}\}_{\ell=1}^{N}$ of N samples and we will represent the distribution $\mu_{k-1}(s_{k-1})\, p(x_k, s_k|s_{k-1})$ by a list of N three-tuples $(\hat{s}_{k-1,\ell}, \hat{x}_{k,\ell}, \hat{s}_{k,\ell})$. From (6.8) and (6.9), we then obtain

$$\lambda_k^{-1} = \int_{s_k}\int_{x_k}\int_{s_{k-1}} \mu_{k-1}(s_{k-1})\, p(x_k, s_k|s_{k-1})\, p(y_k|x_k, s_k, s_{k-1}) ds_k dx_k ds_{k-1} \tag{6.11}$$

$$\approx \frac{1}{N}\sum_{\ell=1}^{N} p_{Y_k|X_k,S_k,S_{k-1}}(y_k|\hat{x}_{k,\ell}, \hat{s}_{k,\ell}, \hat{s}_{k-1,\ell}). \tag{6.12}$$

The recursive computation of (6.8) is then accomplished as follows.

a) Begin with a list $\{\hat{s}_{k-1,\ell}\}_{\ell=1}^{N}$ that represents μ_{k-1}.

b) Extend each particle $\hat{s}_{k-1,\ell}$ to a three-tuple $(\hat{s}_{k-1,\ell}, \hat{x}_{k,\ell}, \hat{s}_{k,\ell})$ by sampling from $p(x_k, s_k|s_{k-1})$.

c) Compute an estimate of λ_k using (6.12).

d) Resampling: draw N samples from the list $\{(\hat{s}_{k-1,\ell}, \hat{x}_{k,\ell}, \hat{s}_{k,\ell})\}_{\ell=1}^{N}$ by choosing each three-tuple with probability proportional to $p_{Y_k|X_k,S_k,S_{k-1}}(y_k|\hat{x}_{k,\ell}, \hat{s}_{k,\ell}, \hat{s}_{k-1,\ell})$.

e) Drop $\hat{s}_{k-1,\ell}$ and $\hat{x}_{k,\ell}$ of each new three-tuple and obtain the new list $\{\hat{s}_{k,\ell}\}_{\ell=1}^{N}$.

Remark 6.1. (Applicability of the particle method)
In Step 2 of the above algorithm, one needs to draw samples from $p(x_k, s_k|s_{k-1})$. A closed-form expression for $p(x_k, s_k|s_{k-1})$ is not required for that. The state transitions may for example be described by a stochastic difference equation (e.g., (2.89)–(2.95)). The required samples are

then generated by "simulating" the stochastic difference equation. For example, simulating the model (2.89)–(2.95) involves drawing samples n_k and z_k from $\mathcal{N}_{0,\sigma_N^2}$ and $\mathcal{N}_{0,\sigma_Z^2}$ respectively (cf. (2.94) and (2.95)). The state transitions may also be modeled by a stochastic differential equation (e.g., (2.59) or (2.82)). First the "continuous" time axis is replaced by a "discrete" time axis, i.e., the stochastic differential equation is converted into a stochastic difference equation (e.g., by the Euler-Maruyama method [84]). Then the resulting stochastic difference equation is simulated. Alternatively, the samples may in principle be obtained by measurements; the particle method may thus even be applied in complete absence of a mathematical model for the state transitions.
The observation model $p_{Y_k|X_k,S_k,S_{k-1}}$, however, has to be available in closed-form (cf. Step 3 and 4).

6.4 A Numerical Example

We consider the random-walk phase model (4.1)–(4.2). The input symbols are i.u.d; the signal constellation is 4-PSK. For this channel, the application of the method of Section 6.3 is straightforward; the results are shown in Figure 6.2. For these computations, we simulated channel input/output sequences of length $n = 10^5$–10^6 and used $N = 10^4$ particles.

Also shown in Figure 6.2 is a tight lower bound on the information rate, which is obtained from using a quantized-phase channel model (with 5000 quantization bins) as an auxiliary channel in the bound of [13]. Note that the results of both approaches practically coincide. Also shown is the information rate for the case where $\sigma_W^2 = 0$, i.e., for the complex AWGN channel with i.u.d. 4-PSK input signals. It is noteworthy that the curves for $\sigma_W^2 = 0$ and $\sigma_W^2 = 0.01$ coincide.

Figure 6.3 depicts the estimate $\hat{I}(X;Y)$ as a function of the number of iterations; we consider ten runs of the particle method and ten runs of the method based on the quantized-phase channel model.

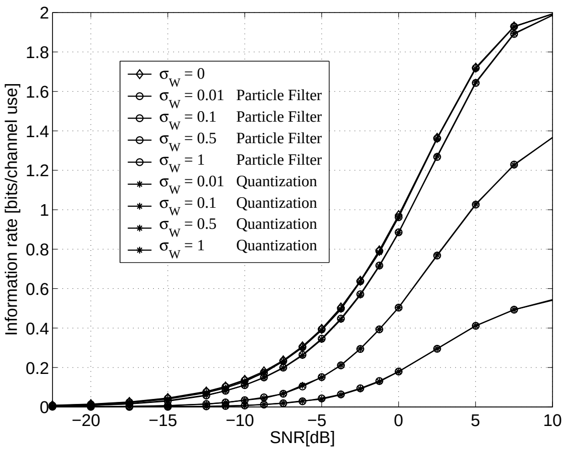


Figure 6.2: Information rates for the channel (4.1)–(4.2) with i.u.d. 4-PSK input symbols.

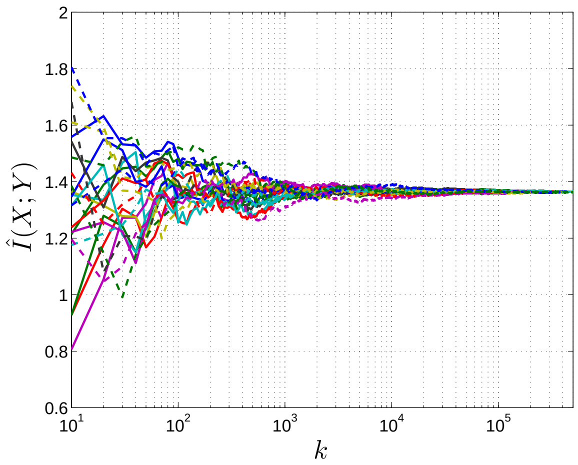


Figure 6.3: $\hat{I}(X;Y)$ as a function of the iteration number k for ten runs of the particle method (dashed) and ten runs of the quantization method (auxiliary-channel bound of [13]) (solid); SNR $=$ 10dB and $\sigma_W = 0.5$ rad.

6.5 Summary

We summarize here the main results of this chapter.

- By using **particle methods**, we have extended the method of [12] to channels with a **continuous state space**. Such methods can also be used to compute the auxiliary-channel bounds of [13]. In contrast to methods based on quantization, particle methods remain practical for **high-dimensional state spaces**.
- The particle methods allow us to compute information rates of state-space models whose state transitions are described by **stochastic differential equations** or **stochastic difference equations**. For such models, the previous methods (e.g., [12]) only lead to *bounds* on the information rates.
- The **accuracy** of our methods depends not only on the sequence length n, but also on the number of particles N.

Chapter 7

Capacity of Continuous Memoryless Channels

We present a numerical method to compute lower and upper bounds on the capacity of continuous memoryless channels. We present numerical results for the Gaussian channel with peak-power and average-power constraints; we outline how the method can be extended to channels with memory such as channels with phase noise (e.g., random-walk phase model). The results of this chapter are based on [39].

7.1 Introduction

We consider the problem of computing lower and upper bounds on the capacity [178][1]

$$C \triangleq \sup_{p(x)} \int_x \int_y p(x)p(y|x) \log \frac{p(y|x)}{p(y)} \triangleq \sup_{p(x)} I(X;Y) \qquad (7.1)$$

of a memoryless channel $p(y|x)$ with input X and output Y, where $p(y) \triangleq \int_x p(x)p(y|x)$. Both X and Y may be discrete or continuous. If x is dis-

[1]Basic notions from information theory are reviewed in Appendix B.

crete, $\int_x g(x)$ stands for the summation of $g(x)$ over its support, otherwise, it stands for integration.

For memoryless channels with finite input alphabets $\mathcal{X}$ and finite output alphabets $\mathcal{Y}$, the capacity (7.1) can be computed by the Blahut-Arimoto algorithm [9] [23]. Recently Matz et al. [125] proposed two modifications of the Blahut-Arimoto algorithm that often converge significantly faster than the standard Blahut-Arimoto algorithm. Vontobel et al. [204] extended the Blahut-Arimoto algorithm to channels with memory and finite input alphabets and state spaces.

For memoryless channels with continuous input and/or output alphabets, the Blahut-Arimoto algorithm is not directly applicable. In this paper, we extend the Blahut-Arimoto algorithm to such channels. It is similar in spirit as the algorithm of Chapter 6 for computing information rates of continuous channels with memory. The key idea is again to represent probability distributions by lists of samples ("particles" or "mass points"; see Section 4.6). In the proposed algorithm, the input distribution $p(x)$ is represented as a particle list $\mathcal{L}_X$. The list $\mathcal{L}_X$ is updated by alternating maximization [187]: first the weights of the particles are updated while the positions of the particles are kept fixed; the Blahut-Arimoto algorithm [9] [23] (or one of the extensions of [125]) can be used for this purpose. Next the positions of the particles are optimized while their weights remain fixed; this can be carried out by several iterations of a gradient method such as steepest ascent or the Newton-Raphson method [19].

The proposed algorithm is related to the algorithm presented in [4]. The latter, however, becomes unstable for certain SNR-values; our algorithm does not suffer from numerical instabilities. Our algorithm is also similar to the algorithm proposed by Chang et al. [31]; also there, the input distribution is represented as a particle list $\mathcal{L}_X$. The algorithm of [31] determines *all* local maxima $\hat{x}_{\max}$ of the relative entropy $D(p(y|x)\|\hat{p}(y))$ after each update of the weights, where $\hat{p}(y)$ is the output distribution corresponding to $\mathcal{L}_X$; those maxima are subsequently appended to the particle list. Finding *all* local maxima of a function is unfortunately often infeasible, especially in high dimensions. In addition, there may be an infinite number of local maxima. The algorithm of [31] is therefore often impractical. Lafferty et al. [105] proposed an alternative algorithm based on Markov-Chain-Monte-Carlo methods (MCMC). The MCMC-based algorithm of [105] is significantly more complex than our algorithm: the complexity of the former grows quadratically in the number of iterations, whereas the complexity of our algorithm depends linearly on the number of iterations; both algorithms seem to converge after about the same number of iterations. In addition, each iteration of the MCMC-based algorithm requires vastly more computations compared to our method.

This chapter is structured as follows. In Section 7.2, we briefly review the Blahut-Arimoto algorithm [9] [23] and two recently proposed alternatives [125]. In Section 7.3, we outline our particle-based algorithm. We present numerical examples in Section 7.4. We summarize the main results of this chapter in Section 7.5. In Section 7.6, we list topics for future research.

7.2 Review of the Blahut-Arimoto Algorithm and Extensions

We start by reviewing the Blahut-Arimoto algorithm [9] [23] and the two extensions of [125]. The Blahut-Arimoto algorithm is an alternating-maximization algorithm [187] for computing the capacity (7.1). One starts with an arbitrary probability (mass or density) function $p^{(0)}(x)$. At each iteration k, the probability function $p^{(k)}(x)$ is updated according to the rule:

$$p^{(k)}(x) = \frac{1}{Z^{(k)}} p^{(k-1)}(x) \exp\Big(D\big(p(y|x) \| p^{(k-1)}(y)\big)\Big), \tag{7.2}$$

where $D\left(q(\cdot)\|p(\cdot)\right)$ is the Kullback-Leibler divergence (or "relative entropy"; cf. Appendix B) defined as

$$D\left(q(\cdot)\|p(\cdot)\right) \triangleq \int_x q(x) \log \frac{q(x)}{p(x)}, \tag{7.3}$$

the expression $p^{(k)}(y)$ is given by

$$p^{(k)}(y) \triangleq \int_{x\in\mathcal{X}} p^{(k)}(x) p(y|x), \tag{7.4}$$

and the factor $Z^{(k)}$ normalizes the probability function $p^{(k)}(x)$:

$$Z^{(k)} \triangleq \int_{x\in\mathcal{X}} p^{(k-1)}(x) \exp\Big(D\big(p(y|x) \| p^{(k)}(y)\big)\Big). \tag{7.5}$$

The mutual information $I^{(k)}$ corresponding to the input probability function $p^{(k)}(x)$ is given by:

$$I^{(k)} \triangleq \int_{x\in\mathcal{X}} p^{(k)}(x) D\big(p(y|x)\|p^{(k)}(y)\big). \tag{7.6}$$

If after n iterations the gap between $I^{(n)}$, which is a *lower* bound on the capacity (7.1), and $\max_{x\in\mathcal{X}} D\big(p(y|x)\|p^{(n)}(y)\big)$, which is an *upper* bound on (7.1), is sufficiently small, i.e., when

$$\max_{x\in\mathcal{X}} D\big(p(y|x)\|p^{(n)}(y)\big) - I^{(n)} < \varepsilon, \tag{7.7}$$

where ε a "small" positive real number (e.g., $\varepsilon = 10^{-5}$), one halts the algorithms and concludes with the estimate $\hat{C} = I^{(n)}$.

Matz et al. [125] proposed two related algorithms for computing the capacity of memoryless channels, i.e., the so-called "natural-gradient-based algorithm" and the "accelerated Blahut-Arimoto algorithm". Both algorithms often converge significantly faster than the standard Blahut-Arimoto algorithm.

In the natural-gradient-based algorithm, the probability function $p^{(k)}(x)$ is recursively updated by the rule [125]

$$p^{(k)}(x) = p^{(k-1)}(x)\left[1 + \mu^{(k)}\cdot\Big(D\big(p(y|x)\|p^{(k-1)}(y)\big) - I^{(k-1)}\Big)\right], \tag{7.8}$$

where $\mu^{(k)}$ is a positive real number ("step size"). Note that $p^{(k)}(x)$ in (7.8) is guaranteed to be normalized. The accelerated Blahut-Arimoto algorithm updates $p^{(k)}(x)$ as [125]

$$p^{(k)}(x) = \frac{1}{Z^{(k)}} p^{(k-1)}(x) \exp\Big(\mu^{(k)}\cdot D\big(p(y|x)\|p^{(n)}(y)\big)\Big), \tag{7.9}$$

where $Z^{(k)}$ is a normalization constant.

Many channels have an associated expense of using each of the input symbols. A common example is the power associated with each input symbol. A constrained channel is a channel with the requirement that the average expense be less than or equal to some specified number $E^{\max}$. The capacity at expense E is defined as [23]

$$C(E) \triangleq \sup_{p(x)\in P_E} \int_x \int_y p(x)p(y|x) \log \frac{p(y|x)}{\int_x p(x)p(y|x)} \triangleq \sup_{p(x)\in P_E} I(X;Y), \tag{7.10}$$

where

$$P_E \triangleq \Big\{p : \mathbb{K}^m \to \mathbb{R} : \int_x p(x) = 1, p(x) \geq 0, E \triangleq \int_x p(x)e(x) \leq E^{\max}\Big\}, \tag{7.11}$$

and $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$. The Blahut-Arimoto algorithm can be extended to constrained channels; the recursion (7.2) is replaced by [23]

$$p^{(k)}(x) = \frac{1}{Z^{(k)}} p^{(k-1)}(x) \exp\Big(D\big(p(y|x) \| p^{(n)}(y)\big) - se(x)\Big), \tag{7.12}$$

where s a positive real number [23]. After n iterations of (7.12), one obtains the following bounds on $C(E)$ [23]:

$$I^{(n)} \leq C(E) \leq \max_{x \in \mathcal{X}} \Big[D\big(p(y|x) \| p^{(n)}(y)\big) - se(x)\Big] + sE^{(n)}, \tag{7.13}$$

where

$$E^{(k)} \triangleq \int_x p^{(k)}(x)e(x). \tag{7.14}$$

Note that $p^{(n)}$ and hence $E^{(n)}$ depend on s. One chooses s so that (1) the condition $E^{(n)} \leq E^{\max}$ is satisfied; (2) the gap between the upper and lower bound (7.16) is as small as possible. To that end, the parameter s may be adjusted after each update (7.12).

When there are multiple constraints $E_j \triangleq \int_x p(x)e_j(x) \leq E_j^{\max}$ ($j = 1, \ldots, L$), the recursion (7.2) is replaced by

$$p^{(k)}(x) = \frac{1}{Z^{(k)}} p^{(k-1)}(x) \exp\Big(D\big(p(y|x) \| p^{(n)}(y)\big) - \sum_{j=1}^{L} s_j e_j(x)\Big) \tag{7.15}$$

and (7.16) is adapted as

$$I^{(n)} \leq C(E) \leq \max_{x \in \mathcal{X}} \left[D\big(p(y|x) \| p^{(n)}(y)\big) - \sum_{j=1}^{L} s_j e_j(x)\right] + \sum_{j=1}^{L} s_j E_j^{(n)}. \tag{7.16}$$

The two Blahut-Arimoto-type algorithms of [125] can similarly be extended to constrained memoryless channels.

7.3 A Particle Method

When X and Y are discrete, the Blahut-Arimoto-type algorithms reviewed in the previous section are practical. Otherwise, the Blahut-Arimoto updates (7.2), (7.8), and (7.9) cannot be carried out as such. We tackle this problem in a straightforward fashion: we represent the input distribution $p(x)$ by a ("long", but finite) particle list

$$\mathcal{L}_X \stackrel{\triangle}{=} \big\{(\hat{x}_1, w_1), (\hat{x}_2, w_2), \ldots, (\hat{x}_N, w_N)\big\}. \tag{7.17}$$

For some channels, the capacity-achieving input distribution is known to be discrete (see e.g., [30] and references therein); the capacity-achieving input distribution can then exactly be represented by a (finite) list $\mathcal{L}_X$. On the other hand, if the capacity-achieving input distribution is continuous, it can obviously not exactly be represented by a (finite) list $\mathcal{L}_X$; nevertheless, it may be well approximated by such a list, especially if N is large (e.g., $N = 10^6$) and the dimension of the input space $\mathcal{X}$ is small.

If one represents the input distribution $p(x)$ by a particle list $\mathcal{L}_X$, the original infinite-dimensional optimization problem (7.1) reduces to the finite-dimensional optimization problem:

$$\hat{C} \stackrel{\triangle}{=} \max_{\hat{x}, w} I(\hat{x}, w), \tag{7.18}$$

where $w \stackrel{\triangle}{=} (w_1, \ldots, w_N)$, $\hat{x} \stackrel{\triangle}{=} (\hat{x}_1, \ldots, \hat{x}_N)$, and the expression $I(\hat{x}, w)$ stands for

$$I(\hat{x}, w) \stackrel{\triangle}{=} \sum_{i=1}^{N} \int_y w_i \, p(y|\hat{x}_i) \log \frac{p(y|\hat{x}_i)}{\hat{p}(y)}. \tag{7.19}$$

The output distribution $\hat{p}(y)$ corresponds to the input distribution $p(x) \stackrel{\triangle}{=} \mathcal{L}_X$, i.e.,

$$\hat{p}(y) \stackrel{\triangle}{=} \sum_{i=1}^{N} w_i \, p(y|\hat{x}_i). \tag{7.20}$$

The mutual information $I(\hat{x}, w)$ (7.19) involves integrals that are often intractable; they may be evaluated by numerical integration (e.g., by the trapezoidal rule) or by Monte-Carlo integration. In the latter approach, the expression (7.19) is computed as:

$$I(\hat{x}, w) = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{M} w_i \log \frac{p\left(\hat{y}_{i,j}|\hat{x}_i\right)}{\hat{p}(\hat{y}_{i,j})}, \tag{7.21}$$

where $\hat{y}_{i,j}$ for $j = 1, \ldots, M$ are samples from the conditional pdfs $p(y|\hat{x}_i)$.

The estimate $\hat{C}$ (7.18) is a lower bound on the capacity (7.1). Note that the mutual information $I(X;Y)$ is concave w.r.t. the distribution $p(x)$, whereas $I(\hat{x}, w)$ is non-convex w.r.t. the positions $\hat{x}$ and the weights w. The list of particles $\mathcal{L}_X^*$ that achieves $\hat{C}$ (7.18) is given by:

$$\mathcal{L}_X^* \triangleq \big\{(\hat{x}_1^*, w_1^*), (\hat{x}_2^*, w_2^*), \ldots, (\hat{x}_N^*, w_N^*)\big\} \triangleq \operatorname*{argmax}_{\hat{x},w} I(\hat{x}, w). \qquad (7.22)$$

The maximization in (7.22) is carried out over the positions $\hat{x}$ and over the weights w. In principle, one can solve (7.22) by alternating maximization [187]:

$$w^{(k)} \triangleq \operatorname*{argmax}_{w} I\big(\hat{x}^{(k-1)}, w\big) \qquad \text{(W-step)} \qquad (7.23)$$

$$\hat{x}^{(k)} \triangleq \operatorname*{argmax}_{\hat{x}} I\big(\hat{x}, w^{(k)}\big) \qquad \text{(X-step)}, \qquad (7.24)$$

where $\hat{x}^{(k)} \triangleq (\hat{x}_1^{(k)}, \ldots, \hat{x}_N^{(k)})$ and $w^{(k)} \triangleq (w_1^{(k)}, \ldots, w_N^{(k)})$ are the positions and weights respectively at the k-th iteration. In the W-step (7.23), the weights w are optimized while the positions $\hat{x}^{(k-1)}$ of the particles are kept fixed. The W-step can be accomplished by the Blahut-Arimoto algorithm [9] [23] or one of the two extensions proposed in [125], since the input alphabet $\mathcal{X} \triangleq \hat{x}^{(k-1)}$ is discrete. In the X-step (7.24), the positions $\hat{x}$ are optimized while the weights $\hat{w}^{(k)}$ of the particles remain fixed; this optimization problem is non-convex and often difficult. Therefore, instead of performing the maximization (7.24), we select the positions $\hat{x}^{(k)}$ such that

$$D(p(y|x_i^{(k)}) \| p^{(k)}(y)) \geq D(p(y|x_i^{(k-1)}) \| p^{(k-1/2)}(y)), \qquad (7.25)$$

for $i = 1, \ldots, N$ with

$$\hat{p}(y)^{(k-1/2)} \triangleq \sum_{i=1}^{N} w_i^{(k)}\, p(y|\hat{x}_i^{(k-1)}). \qquad (7.26)$$

From (7.25) it follows:

$$I\big(\hat{x}^{(k)}, w^{(k)}\big) \geq I\big(\hat{x}^{(k-1)}, w^{(k)}\big). \qquad (7.27)$$

Positions $\hat{x}^{(k)}$ that fulfill the condition (7.25) can be obtained by means of a gradient method as, e.g., steepest ascent or the Newton-Raphson method [19]. In steepest ascent for example, the alphabet $\hat{x}$ is updated according to the rule:

$$\hat{x}_i^{(k)} = \hat{x}_i^{(k-1)} + \lambda_k \, \frac{\partial}{\partial \hat{x}_i} D(p(y|\hat{x}_i)\|\hat{p}(y))\bigg|_{\hat{x}^{(k-1)},w^{(k)}}, \tag{7.28}$$

where λ_k ("step size") is a positive real number that in general depends on k; note that the pdf $\hat{p}(y)$ (cf. (7.20)), which appears in the RHS of (7.28), depends on $\hat{x} \triangleq (\hat{x}_1, \ldots, \hat{x}_N)$.

By adapting the step size λ_k according to the Armijo rule [19], one can guarantee (7.25) and hence (7.27). The partial derivative occurring in the RHS of (7.28) can be evaluated as:

$$\begin{aligned}\frac{\partial}{\partial \hat{x}_i} D(p(y|\hat{x}_i)\|\hat{p}(y)) = \int_y p(y|\hat{x}_i) \frac{\partial}{\partial \hat{x}_i} \log p(y|\hat{x}_i) \\ \cdot \left(1 + \log \frac{p(y|\hat{x}_i)}{\hat{p}(y)} - w_i \frac{p(y|\hat{x}_i)}{\hat{p}(y)}\right).\end{aligned} \tag{7.29}$$

If the integral in the RHS of (7.29) is not available in closed-form, it may be computed by means of numerical integration or Monte-Carlo integration. In the update rule (7.29), the positions $\hat{x}^{(k)}$ are obtained from the positions $\hat{x}^{(k-1)}$ by a single steepest-ascent step; obviously, the positions $\hat{x}^{(k)}$ may also be determined by multiple subsequent steepest-ascent steps.

In summary, we propose the following algorithm:

a) Start with an initial list of mass points

$$\mathcal{L}^{(0)} \triangleq \big\{(\hat{x}_1^{(0)}, w_1^{(0)}), (\hat{x}_2^{(0)}, w_2^{(0)}), \ldots, (\hat{x}_N^{(0)}, w_N^{(0)})\big\},$$

with uniform weights $w_i^{(0)} = 1/N$ for $i = 1, \ldots, N$, where N is sufficiently large.

b) W-step:
Determine the new weights $w^{(k)}$ by solving (7.23); this can be done by means of the Blahut-Arimoto algorithm [9] [23] or one of the extensions of [125].

c) X-step:
While the weights $w^{(k)}$ remain fixed, move the points $\hat{x}^{(k-1)}$ in order to increase $I\big(\hat{x}^{(k-1)}, w^{(k)}\big)$ (e.g., by means of a gradient method), which results in a new set of points $\hat{x}^{(k)}$.

d) Iterate 2–3.

After n iterations, a lower bound $L^{(n)}$ for the capacity (7.1) is eventually obtained as:

$$C \geq L^{(n)} \triangleq I\big(\hat{x}^{(n)}, w^{(n)}\big). \tag{7.30}$$

An upper bound on the capacity (7.1) is given by

$$U^{(n)} \triangleq \max_{x \in \mathcal{X}} D\big(p(y|x) \| p^{(n)}(y)\big) \geq C, \tag{7.31}$$

where $p^{(n)}(y)$ is the output distribution corresponding to the input distribution $p(x) \triangleq \mathcal{L}_X^{(n)}$ (cf. (7.20)).

Since the sequence $I^{(k)}$ is non-decreasing, i.e., $I^{(k+1)} \geq I^{(k)}$ and mutual information is bounded from above, the algorithm converges to a local maximum of $I(x, w)$. The algorithm may converge to the capacity-achieving input distribution if the latter is discrete, especially if the number N of particles is sufficiently large, the dimension of the input space $\mathcal{X}$ is small, and the initial list $\mathcal{L}^{(0)}$ sufficiently covers the input space $\mathcal{X}$. Convergence to the capacity-achieving input distribution is only guaranteed, however, in the limit of an infinite number of particles.

The proposed algorithm can straightforwardly be extended to constrained channels (cf. Section 7.2): in the W-step, one iterates the recursion (7.12) (or one of the extensions of [125]); after each W-step, one may adjust the parameter s (cf. (7.12)). In the X-step, one moves the points $\hat{x}^{(k-1)}$ in order to increase $I\big(\hat{x}^{(k-1)}, w^{(k)}\big) + sE\big(\hat{x}^{(k-1)}, w^{(k)}\big)$ (e.g., by means of a gradient method), where

$$E\big(\hat{x}, w\big) \triangleq \sum_{i=1}^{N} w_i e(\hat{x}_i). \tag{7.32}$$

After n iterations, one obtains the following lower bound on the capacity $C(E)$ (7.10):

$$C(E) \geq L^{(n)}(E) \triangleq I\big(\hat{x}^{(n)}, w^{(n)}\big). \tag{7.33}$$

An upper bound on $C(E)$ (7.10) is given by:

$$U^{(n)}(E) \triangleq \max_{x \in \mathcal{X}} \Big[D\big(p(y|x) \| p^{(n)}(y) - se(x)\big) \Big] + sE\big(\hat{x}^{(n)}, w^{(n)}\big) \geq C(E). \tag{7.34}$$

7.4 Numerical Example: Gaussian Channel

By means of the above algorithm, we computed the capacity of a Gaussian channel described by the equation

$$Y_k = X_k + N_k, \tag{7.35}$$

where $X_k \in \mathbb{R}$, and N_k is an i.i.d. zero-mean real Gaussian random variable, independent of X_k, with (known) variance σ_0^2. We considered the average-power constraint $\mathrm{E}[X^2] \leq P$ and two different peak-power constraints, i.e., $\Pr[|X| > A] = 0$ and $\Pr[0 \leq X \leq A] = 1$.

In the simulations, we used 100 particles $\hat{x}$ and between 100 and 1000 alternating-maximization iterations;[2] each such iteration consisted of 1000 accelerated Blahut-Arimoto iterations (W-step) and 20 steepest descent updates (X-step). Our experiments have shown that, for the problem at hand, the accelerated Blahut-Arimoto algorithm converges faster than the natural-gradient based Blahut-Arimoto algorithm [125]; we optimized the constant step size $\mu^{(k)} \triangleq \mu$ (for all k) of the accelerated Blahut-Arimoto algorithm (cf. (7.9)), resulting in the value $\mu = 3$ (for all k). We compute $I(\hat{x}, w)$ (7.19) by numerical integration.

In order to assess the performance of our method, we considered a channel for which the capacity and corresponding input distribution are available in closed-form, i.e., the Gaussian channel (7.35) with average-power constraint $\mathrm{E}[X^2] \leq P$. As is well known, the capacity C (in bits per channel use) of that channel is given by [178]:

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_0^2} \right). \tag{7.36}$$

Fig. 7.1(a) shows the expression (7.36) (for $P = 1$) together with the lower bound $L^{(n)}(E)$ (7.33), where the signal-to-noise ratio (SNR) is

[2]The number of required iterations increases with the SNR.

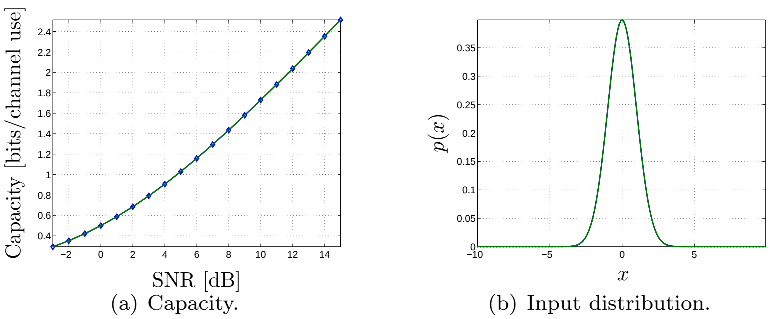


Figure 7.1: Gaussian channel with constraint $\mathrm{E}[X^2] \leq 1$.

defined as

$$\mathrm{SNR[dB]} \triangleq 10 \log_{10} \left(\frac{P}{\sigma_0^2} \right). \tag{7.37}$$

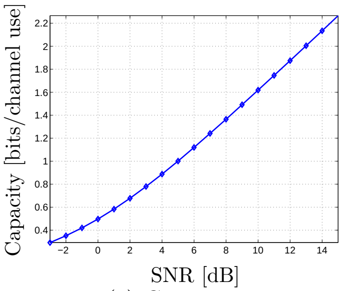
Also shown in Fig. 7.1(a) is an approximation $\hat{U}^{(n)}(E)$ of the upper bound $U^{(n)}(E)$. Since the input alphabet $\mathcal{X}$ is unbounded, the upper bound $U^{(n)}(E)$ (7.34) is intractable; the estimate $\hat{U}^{(n)}(E)$ of $U^{(n)}(E)$ is obtained by restricting the maximization (7.34) to the interval [-10,10], i.e.,

$$\hat{U}^{(n)}(E) \triangleq \max_{x \in [-10,10]} \Big[D\big(p(y|x) \| p^{(n)}(y)\big) - se(x) \Big] + sE\big(\hat{x}^{(n)}, w^{(n)}\big). \tag{7.38}$$

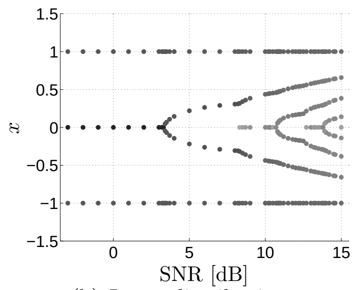
It can be seen from Fig. 7.1(a) that $\hat{U}^{(n)}(E)$ and $L^{(n)}(E)$ are practically equal to the capacity C. The deviation between $L^{(n)}$, $\hat{U}^{(n)}$ and C is about 10^{-5} bits/channel use. The accuracy could in fact be improved by increasing the number of particles and iterations.

It is well known that the capacity-achieving input distribution is a zero-mean Gaussian distribution $\mathcal{N}(0, P)$ with variance P [178]. In Fig. 7.1(b), this distribution is shown together with the particle-based approximation (for $P = 1$). Again, the exact and the numerical results practically coincide.

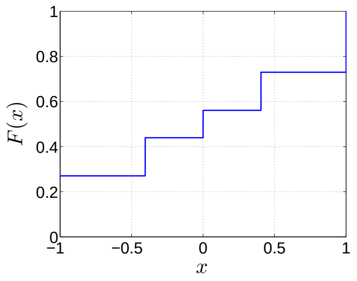
Fig. 7.2 shows the results for Gaussian channel with average-power constraint $\mathrm{E}[X^2] \leq P$ and peak-power constraint $\Pr[|X| > A] = 0$ ($A = 1$ and $P = 0.5$). Fig. 7.2(a) shows the lower bound $L^{(n)}$ and upper bound $U^{(n)}$ on the channel capacity as a function of the SNR (7.37). It can be seen from Fig. 7.2(a) that both bounds coincide. In Fig. 7.2(b), the capacity-achieving input distributions are depicted: the dots are the constellation points, their probability mass is encoded in the grayscale (white: $p = 0$; black: $p = 0.5$). The capacity-achieving input distribution for this channel is discrete (see e.g., [30] and references therein). Note that our algorithm does not make use of that fact: it has to determine both the number and the position of the mass points of the capacity-achieving input distribution. As an illustration, the capacity-achieving cumulative input distribution $F(x)$ is depicted in Fig. 7.2(c) for SNR = 13dB. Fig. 7.2(d) shows how the particles explore the input space during the iterations: initially, the particles are uniformly distributed in the interval $[-1, 1]$; they gradually move towards the signal points of the capacity-achieving input distribution (cf. Fig. 7.2(c)).



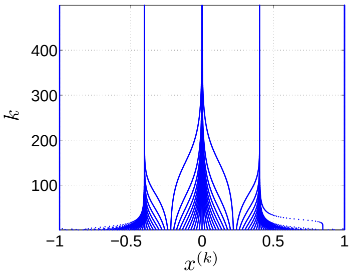
(a) Capacity.



(b) Input distribution.



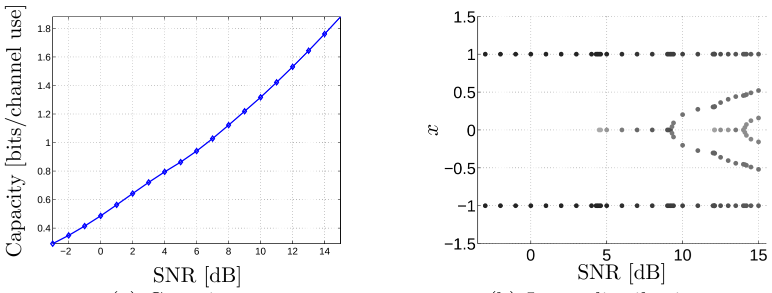
(c) Cumulative input distribution at SNR=13dB.



(d) Particles.

Figure 7.2: Gaussian channel with constraints $\mathrm{E}[X^2] \leq 0.5$ and $\Pr[|X| > 1] = 0$.

Fig. 7.3 shows the results for the Gaussian channel with peak-power constraint $\Pr[|X| > A] = 0$ (with $A = 1$). Fig. 7.3(a) shows the lower bound $L^{(n)}$ and upper bound $U^{(n)}$ on the channel capacity as a function of the SNR, which is defined as $\mathrm{SNR[dB]} \triangleq 10\log_{10}\left(\frac{A^2}{\sigma_0^2}\right)$; both bounds again coincide. Fig. 7.3(b) shows the corresponding input probability mass functions.



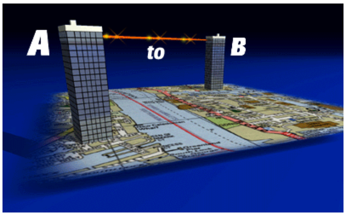
(a) Capacity. (b) Input distribution.

Figure 7.3: Gaussian channel with constraint $\Pr[|X| > 1] = 0$.

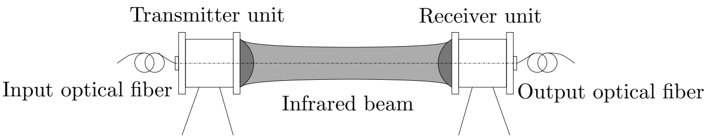
Fig. 7.4 shows the results for the Gaussian channel with peak-power constraints $\Pr[0 \leq X \leq A] = 1$ (with $A = 1$). This channel is a simple model for free-space optical communication. Before we discuss the results, we briefly elaborate on free-space optical communication. (We will closely follow [140].) Free-space optical communication systems are nowadays commercially available (see [161], and references therein); they are mostly used for short metropolitan links, as illustrated in Fig. 7.4(a). The transmission of the input signal is performed by light emitting diodes (LED) or laser diodes (LD), as depicted in Fig. 7.4(b) [161]; conventional diodes emit light in the infrared spectrum. The main characteristic of a free-space optical communications channel is that the impact by noise is mostly due to ambient light, other noise sources can often be neglected; therefore, the noise can be assumed to be independent of the input. The direct line-of-sight path is often dominant, as a consequence, the impact of inter-symbol interference due to multi-path propagation can be neglected. Due to eye safety and the danger of potential thermal skin damage, the optical peak-power has to be constrained.[3]

[3]See, e.g., `http://www.lasermet.com/resources/classification-overview.html`.

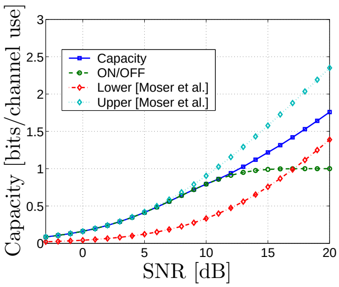
Fig. 7.4(c) shows (1) the lower bound $L^{(n)}$ and upper bound $U^{(n)}$ on the channel capacity, which, as before, coincide; (2) analytical upper and lower bounds by Moser et al. [140][4]; (3) the information rates corresponding to the input constellation $\mathcal{X} = \{0, 1\}$ ("ON/OFF keying"), which is frequently used in practical free-space optical communications systems [161]. In Fig. 7.4(d), the capacity-achieving input constellations are depicted. Note that ON/OFF keying achieves capacity for SNR-values below 11dB, which are realistic SNR-values [161].



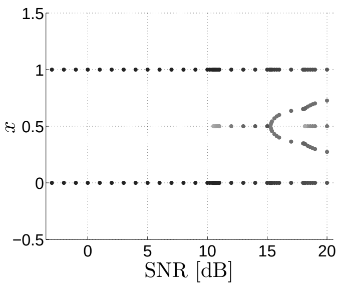
(a) Application.



(b) Communications system [161].



(c) Capacity.



(d) Input distribution.

Figure 7.4: Gaussian channel with $\Pr[0 \leq X \leq 1] = 1$.

[4]We have plotted the low-power bounds proposed in [140]. The high-power lower bound of [140] is for $A = 1$ looser than the low-power lower bound. The high-power upper bound only holds for large values of A, it is trivial for $A = 1$.

7.5 Summary

- We proposed a simple **numerical algorithm** to compute (lower and upper bounds on) the capacity of **continuous memoryless channels**. The input distribution is represented by a **finite list of mass points**; the list is computed by **alternating maximization**: first, the weights of the mass points are updated by means of the Blahut-Arimoto algorithm (with the current positions of the mass points as input alphabet), then, the positions of the mass points are optimized (while the weights of the mass points are fixed); this non-convex optimization problem is handled by gradient methods.
- We have presented numerical results for the **Gaussian channel** with average-power and/or peak-power constraints. The algorithm seems to lead to accurate results both for **discrete** and for **continuous** capacity-achieving input distributions. Note that the algorithm does not need to know **in advance** whether the capacity-achieving input distribution is discrete or continuous.
- The algorithm is **not guaranteed** to converge to the capacity-achieving input distribution, in practice however, it usually does if the capacity-achieving input distribution is **discrete**.

7.6 Outlook

- The methods of this chapter can be extended to channels **with memory**, e.g., inter-symbol-interference channels (ISI). The general idea is again simple: one interleaves the Blahut-Arimoto-type algorithm of [204] (W-step) with gradient ascent (X-step). The resulting algorithm involves forward and backward sweeps in which sum-product messages and *gradients* of sum-product messages are computed (cf. Section 4.8.1). We have implemented such algorithms and we obtained promising preliminary results. Further investigation of those algorithms is needed.
- It would be interesting to extend our methods to the computation of rate distortion functions [178].

Chapter 8

Analog Electronic Circuit for PN-Synchronization

This chapter is not directly related to the problem of carrier-phase estimation. However, it is strongly related to (1) the problem of synchronization, more specifically, pseudo-noise (PN) synchronization; (2) message passing on factor graphs, in particular, the implementation of message-passing algorithms as dynamical systems (analog electrical circuits).

More precisely, we present a clockless low-power analog circuit that synchronizes to pseudo-noise (PN) signals. The circuit operates in continuous time and does not contain a digital clock; in other words, it avoids the problem of timing synchronization.

The circuit is derived by translating a discrete-time message-passing algorithm into continuous time. The circuit exhibits the phenomenon of entrainment—we thus establish a connection between entrainment and statistical state estimation.

The results of this chapter are based on joint work with Matthias Frey, Neil Gershenfeld, Tobias Koch, Patrick Merkli and Benjamin Vigoda [201]. My personal contribution concerns the statistical estimation aspect, and not the hardware implementation nor measurement of the circuit.

8.1 Introduction

Pseudo-random signals play an important role in spread-spectrum communications [182] [139] in various measurement systems. In such systems, the synchronization of pseudo-random ("pseudo noise") signals is a problem of significant interest. The standard solution to this problem is based on correlating the incoming signal with (a segment of) the pseudo-random signal, which leads to a long acquisition time if the period of the signal is large.

Perhaps the most popular class of pseudo-random signals are generated by linear-feedback shift registers (LFSRs). Both Gershenfeld and Grinstein [74] and Yang and Hanzo [219] [220] observed that LFSR sequences can be synchronized by means of a "soft" or "analog" LFSR. The approach of [74] is system theoretic: the soft LFSR is a dynamical system with entrainment capabilities (cf. [158] [38] [3]) obtained by embedding the discrete state space of the LFSR into a continuous state space. In contrast, the (better) soft LFSR of [219] [220], which was independently obtained also in [50], is derived from statistical estimation; it achieves quick synchronization—e.g., after 150 samples at 0 dB for an LFSR with a period of 2^{15}–1 samples—at very low computational cost. Related algorithms, some of them more complex and more powerful, were presented in [50] [51] [133] [32] [224].

Here, we connect the dynamical systems view of [74] with the statistical view of [219] [220], both in discrete time and in continuous time. First, we derive the soft LFSR of [219] [220] as forward-only message passing in the corresponding factor graph. We then propose a new continuous-time analog of both the LFSR and the soft LFSR, both suitable for realization as electronic circuits. We actually implemented one such circuit, and we report some measurements. It is thus demonstrated that continuous-time dynamical systems (such as clockless electronic circuits) with good entrainment properties can be derived from message passing algorithms for statistical state estimation. Such systems/circuits may have substantial advantages in terms of speed and/or power consumption over digital implementations in some applications, and they may enable entirely new applications. However, such applications are outside the scope of this thesis.

This chapter is organized as follows. We begin by stating the discrete-time problem in Section 8.2. In Section 8.3, we review maximum-likelihood estimation and its interpretation as forward-only message passing in a cycle-free factor graph. In Section 8.4, we obtain the soft LFSR as forward-only message passing through another factor graph, and we present some simulation results. A continuous-time analog of the (discrete-time) LFSR is proposed in Section 8.5. The corresponding continuous-time analog of the soft LFSR and its realization as an electronic circuit are described in Section 8.6. Some measurements of this circuit are reported in Section 8.7. In Section 8.8, we summarize the main results of this chapter; in Section 8.9, we outline several topics for future research. Some details of alternative versions of the soft LFSR (sum-product, max-product, and Gershenfeld-Grinstein) are given in Appendix L.

8.2 Noisy LFSR Sequences

For fixed integers ℓ and m satisfying $1 \leq \ell < m$, let

$$X \triangleq X_{-m+1}, \ldots, X_{-1}, X_0, X_1, X_2, \ldots \tag{8.1}$$

be a binary sequence satisfying the recursion

$$X_k = X_{k-\ell} \oplus X_{k-m} \tag{8.2}$$

for $k = 1, 2, 3, \ldots$, where "$\oplus$" denotes addition modulo 2. Any such sequence will be called a LFSR (linear-feedback shift register) sequence. For $k \geq 0$, the m-tuple $[X]_k \triangleq (X_{k-m+1}, \ldots, X_{k-1}, X_k)$ will be called the *state* of X at time k. The sequence $X_1, X_2, \ldots$ is observed via a memoryless channel with transition probabilities $p(y_k|x_k)$. The situation is illustrated in Fig. 8.1 for $\ell = 1$ and $m = 3$; the boxes labeled "D" are unit-delay cells.

Note that the restriction to two right-hand terms ("taps") in (8.2) is made only to keep the notation as simple as possible; all results of this paper are easily generalized to more taps. We also remark that, in most applications (and in our examples), LFSR sequences with the maximal period of $2^m - 1$ are preferred, but this condition plays no essential role in this paper.

From the received sequence $Y_1, Y_2, \ldots, Y_n$, we wish to estimate the state $[X]_n$ of the transmitted sequence. The computation of the maximum-

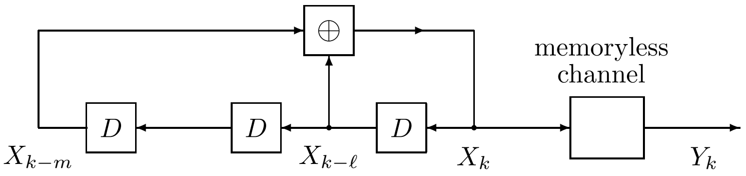


Figure 8.1: LFSR sequence observed via a noisy channel.

likelihood (ML) estimate is straightforward and well known [182]; however, the complexity of this computation is proportional to $n2^m$, which makes it impractical unless m is small.

In the examples, we will assume that the channel is defined by

$$Y_k = \tilde{X}_k + Z_k \tag{8.3}$$

with

$$\tilde{X}_k \triangleq \begin{cases} 1, & \text{if } X_k = 0 \\ -1, & \text{if } X_k = 1 \end{cases} \tag{8.4}$$

(i.e., binary antipodal signaling) and where $Z = Z_1, Z_2, \ldots$ is white Gaussian noise (i.e., independent zero-mean Gaussian random variables) with variance σ^2.

8.3 ML Estimation, Trellis, and Factor Graphs

Let us recall some basic facts. First, we note that the mapping $x \mapsto [x]_k$ (from sequences to states) is invertible for any $k \geq 0$: from the forward recursion (8.2) and the backward recursion $X_{k-m} = X_k \oplus X_{k-\ell}$, the complete sequence x is determined by its state at any time k.

Second, we consider the maximum-likelihood (ML) estimate of $[X]_n$. Using the notation $y^n \triangleq (y_1, \ldots, y_n)$ and $x^n \triangleq (x_{-m+1}, \ldots, x_n)$, the ML estimate of $[X]_n$ is the maximum (over all possible states $[x]_n$) of

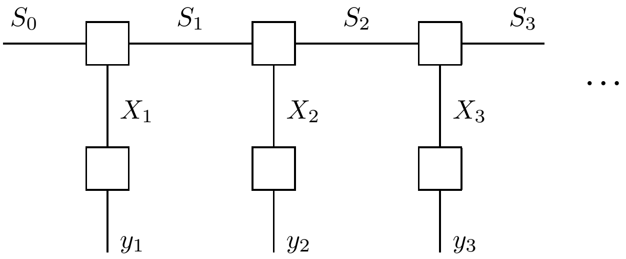


Figure 8.2: Factor graph (Forney-style) corresponding to the trellis of the system in Fig. 8.1.

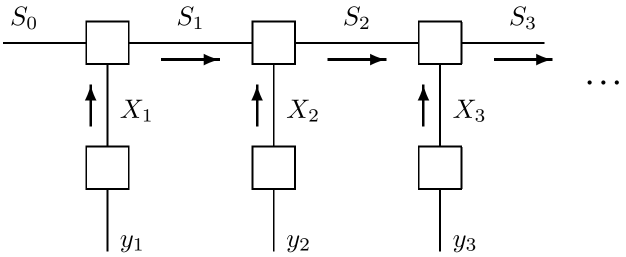


Figure 8.3: Forward-only message passing through the factor graph of Fig. 8.2.

the likelihood function

$$p(y^n \mid [x]_n) = p(y^n|x^n) \tag{8.5}$$

$$= \prod_{k=1}^{n} p(y_k|x_k). \tag{8.6}$$

For the channel (8.3), maximizing (8.6) amounts to maximizing the correlation between $\tilde{x}^n$ and y^n.

Third, we note that the computation of (8.6) may be viewed as the forward recursion of the BCJR algorithm [15] through the trellis of the system or—equivalently—as forward-only message passing through the corresponding factor graph. Let us consider this more closely.

A factor graph of our system is shown in Fig. 8.2. The nodes in the top row of Fig. 8.2 represent $\{0,1\}$-valued functions $J(s_{k-1}, x_k, s_k)$ that indicate the allowed combinations of old state $s_{k-1} = [x]_{k-1}$, output symbol x_k, and new state $s_k = [x]_k$. The nodes in the bottom row of Fig. 8.2 represent the channel transition probabilities $p(y_k|x_k)$. As a whole, the factor graph of Fig. 8.2 represents the function

$$p(y^n|x^n)J(x^n, s^n) = \prod_{k=1}^{n} J(s_{k-1}, x_k, s_k)\, p(y_k|x_k) \tag{8.7}$$

(defined for arbitrary binary sequences x^n), where

$$J(x^n, s^n) \triangleq \prod_{k=1}^{n} J(s_{k-1}, x_k, s_k) \tag{8.8}$$

is the indicator function of valid LFSR sequences, which may also be viewed as a uniform prior over all valid x^n.

It then follows from basic factor graph theory [103] [119] that the a posteriori probability distribution over $S_n = [X]_n$ (and thus the MAP / ML estimate of S_n) is obtained from forward-only sum-product message passing as illustrated in Fig. 8.3. Since the trellis has no merging paths, the sum-product rule for the computation of messages reduces to a product-only rule and coincides with the max-product rule. By taking logarithms, the product-only rule becomes a sum-only rule; for the channel (8.3), this amounts to a recursive computation of the correlation between $\tilde{x}^n$ and y^n.

8.4 The Soft LFSR

Another factor graph for our system is shown (for $\ell = 1$ and $m = 3$) in Fig. 8.4. This factor graph represents the function

$$p(y^n|x^n)J(x^n) = \prod_{k=1}^{n} \delta[x_k \oplus x_{k-\ell} \oplus x_{k-m}]\, p(y_k|x_k), \tag{8.9}$$

where $\delta[.]$ is the Kronecker delta and where $J(x^n) = \prod_{k=1}^{n} \delta[x_k \oplus x_{k-\ell} \oplus x_{k-m}]$ is the indicator function for valid LFSR sequences according to (8.2).

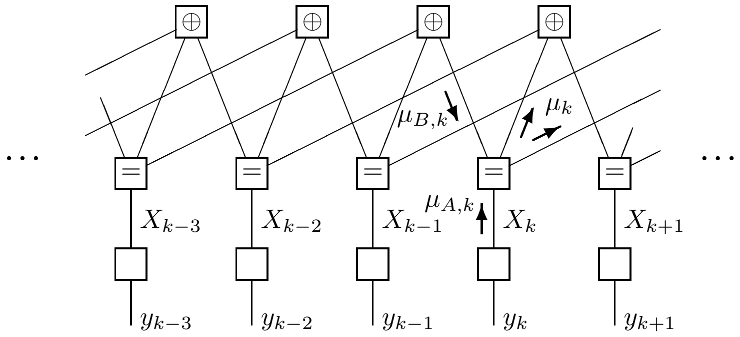


Figure 8.4: Factor graph corresponding directly to Fig. 8.1.

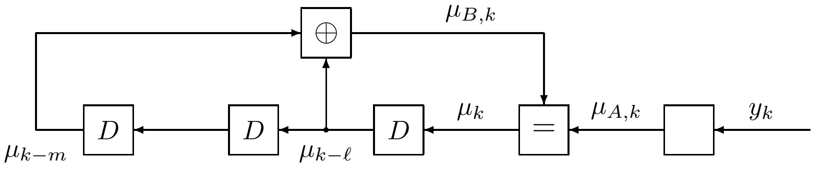


Figure 8.5: Computation of messages in Fig. 8.4 by a "soft LFSR".

As this factor graph has cycles, the standard sum-product and max-product algorithms become iterative algorithms. Such algorithms were investigated in [32] and [224]. Here, however, we stick to (non-iterative) forward-only message passing. Since (full-state) forward-only message passing is optimal in Fig. 8.3, there is hope that (scalar) forward-only message passing in Fig. 8.4 might do well also. In any case, forward-only message passing in Fig. 8.4 amounts to a simple recursion, which may be interpreted as running the received sequence Y through the "soft LFSR" circuit of Fig. 8.5. The quantities $\mu_{A,k}$, $\mu_{B,k}$, and μ_k in Fig. 8.5 are the messages indicated in Fig. 8.4. Note that the same message μ_k is sent along two edges out of the equality check node corresponding to X_k.

The computation of these messages (as indicated in Fig. 8.5) is a standard application of the sum-product or max-product rules. Each message represents "pseudo-probabilities" $\tilde{p}(0)$ and $\tilde{p}(1)$, e.g., in the form

$\tilde{p}(0)/\tilde{p}(1)$ or $\tilde{p}(0) - \tilde{p}(1)$. For the latter representation, the explicit sum-product update rules are as follows:

Initialization: $\mu_k = 0$ for $k = -m+1, -m+2, \ldots, 0$.

Recursion (for $k = 1, 2, 3, \ldots$):

$$\mu_{A,k} = \frac{p(y_k|x_k = 0) - p(y_k|x_k = 1)}{p(y_k|x_k = 0) + p(y_k|x_k = 1)} \tag{8.10}$$

$$\overset{\text{for AWGN}}{=} \frac{\exp(2y_k/\sigma^2) - 1}{\exp(2y_k/\sigma^2) + 1} \tag{8.11}$$

$$\mu_{B,k} = \mu_{k-\ell} \cdot \mu_{k-m} \tag{8.12}$$

$$\mu_k = \frac{\mu_{A,k} + \mu_{B,k}}{1 + \mu_{A,k} \cdot \mu_{B,k}} \tag{8.13}$$

Equation (8.10) holds for a general memoryless channel while (8.11) is the specialization to the channel specified at the end of Section 8.1. At any given time k, an estimate of X_k is obtained as

$$\hat{X}_k \triangleq \begin{cases} 0, & \text{if } \mu_k \geq 0 \\ 1, & \text{if } \mu_k < 0 \end{cases} \tag{8.14}$$

and $[\hat{X}]_k = (\hat{X}_{k-m+1}, \ldots, \hat{X}_{k-1}, \hat{X}_k)$ is an estimate of the state $[X]_k$.

The sum-product update rules for the case where the messages represent the ratio $\tilde{p}(0)/\tilde{p}(1)$ are given in the appendix together with the max-product rules and the analog LFSR of [74].

Simulation results for maximum-length LFSR sequences with memory $m = 15$ and $m = 31$ are given in Figures 8.6–8.8. All these figures show plots of the probability of synchronization

$$P_{\text{synch}}(k) \triangleq P\Big([\hat{X}]_k = [X]_k\Big) \tag{8.15}$$

either vs. the time index k or vs. the signal-to-noise ratio $1/\sigma^2$ where σ^2 is the noise variance.

As is obvious from these plots (and from similar plots in [219] [220] [50]) the soft LFSR quickly achieves synchronization for sufficiently low noise power (up to about 0 dB) but fails for high noise power. It is remarkable

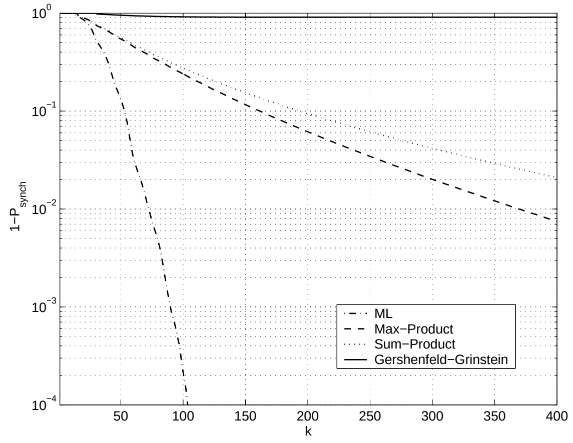


Figure 8.6: $1 - P_{\text{sync}}(k)$ for the LFSR with $m = 15$ ($\ell = 1$) at SNR $= 0$ dB. Algorithms (in the order of increasing performance): G.-G. soft LFSR [74]; sum-product soft LFSR; max-product soft LFSR; maximum likelihood (ML).

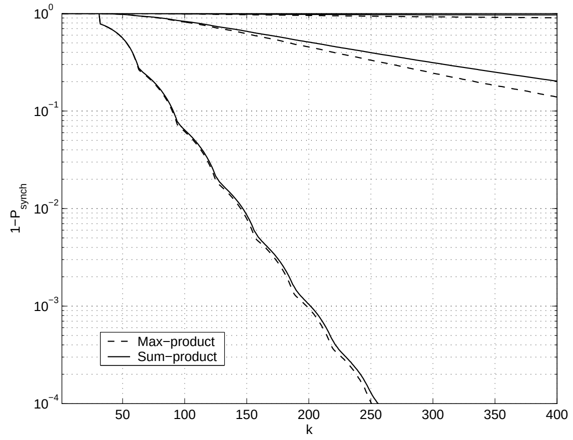


Figure 8.7: $1 - P_{\text{sync}}(k)$ for the LFSR with $m = 31$ ($\ell = 3$) for three different signal-to-noise ratios: SNR = -2.92 dB ($\sigma = 1.4$), SNR $= 0$ dB ($\sigma = 0$), and SNR $= 4.44$ dB ($\sigma = 0.6$). Algorithms: max-product soft LFSR and sum-product soft LFSR.

that the max-product algorithm gives better performance than the sum-product algorithm, but the difference is small.

We also note that better performance can be achieved both with more complex forward-only message passing [50] [51] and with iterative message passing, cf. [32] [224].

8.5 A Continuous-Time Pseudo-Random Generator

We now proceed to an analog of Figures 8.1 and 8.5 in continuous time. Our proposal for a continuous-time analog of Fig. 8.1 is shown in Fig. 8.9. The signal $X(t)$ in Fig. 8.9 takes values in the set $\{+1, -1\}$. The multiplier in Fig. 8.9 corresponds to the mod-2 addition in Fig. 8.1.

How should we translate the delay cells in Fig. 8.1 to continuous time? An obvious approach would be to simply translate them into continuous-time delay cells. However, ideal continuous-time delay cells cannot be realized by real circuits (except perhaps in optics); even a delay line (e.g., a piece of wire) has a low-pass characteristic.

We therefore choose to replace the discrete-time delay cells of Fig. 8.1 by low-pass filters with transfer functions $H_1(s)$ and $H_2(s)$ as shown in Fig. 8.9. Since the output signal of such filters is not restricted to $\{+1, -1\}$, we introduce threshold elements between the filter outputs and the multiplier, which reduce the filtered signals to their sign ($+1$ or -1). These threshold elements have no counterpart in Fig. 8.1 (and will create a small problem in the receiver).

The memoryless channel in Fig. 8.1 is translated into the additive white Gaussian channel shown in Fig. 8.9.

The type of signal $X(t)$ generated by the circuit of Fig. 8.9 is illustrated in Fig. 8.10 (top). From our simulations, it appears that the signal $X(t)$ is generically periodic. The actual signal depends, of course, on the two filters. In our examples, the first filter (with transfer function $H_1(s)$) is a 5-th order Butterworth filter with -3 dB frequency 1.6 kHz, and the second filter (with transfer function $H_2(s)$) is a cascade of 6 such filters. With these filters, the circuit of Fig. 8.9 is a dynamical system with a

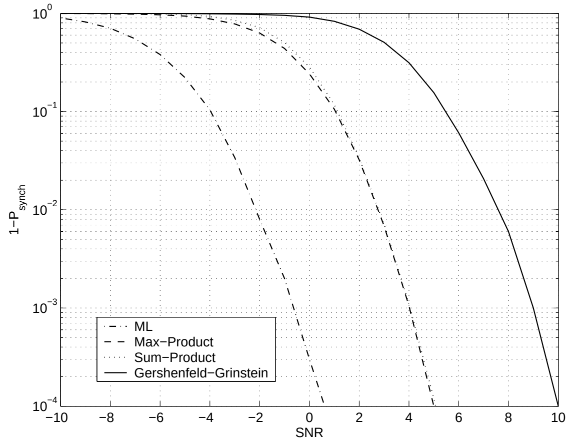


Figure 8.8: $1 - P_{\text{sync}}(k = 100)$ vs. SNR for the LFSR with $m = 15$ ($\ell = 1$). Algorithms (in the order of increasing performance): G.-G. soft LFSR [74]; sum-product soft LFSR; max-product soft LFSR; maximum likelihood (ML).

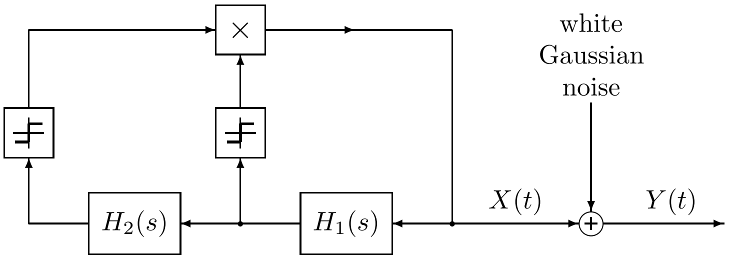


Figure 8.9: Continuous-time analog to Fig. 8.1 with low-pass filters instead of delay cells.

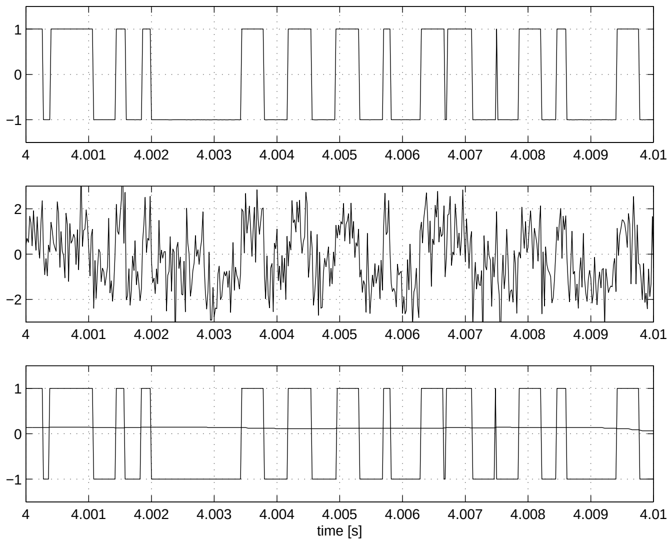


Figure 8.10: Top: example of pseudo-random signal $X(t)$ generated by the circuit of Fig. 8.9. Middle: noisy signal $Y(t)$ as in Fig. 8.9 at SNR = 0 dB. Bottom: measured output signal $\hat{X}(t)$ of the circuit of Fig. 8.11 fed with $Y(t)$.

35-dimensional state space. The resulting signal $X(t)$ is periodic with a period of 34 ms, 10 ms of which are shown in Fig. 8.10 (top).

It should be emphasized that, at present, we do not have a theory of such circuits and we cannot predict the period of the generated sequence $X(t)$. However, our simulation experiments (e.g., in [99] [42] [43]) suggest that a long period—"long" meaning many zero-crossings—requires a high-dimensional state space.

8.6 A Circuit that Locks onto the Pseudo-Random Signal

A continuous-time analog to the soft LFSR of Fig. 8.5 matched to the pseudo-random generator of Fig. 8.9 is shown in Fig. 8.11. The linear

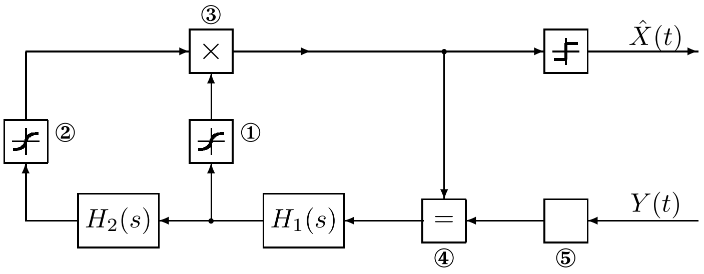


Figure 8.11: Continuous-time analog of Fig. 8.5.

filters $H_1(s)$ and $H_2(s)$ in Fig. 8.11 are identical to those in Fig. 8.9. All signals in Fig. 8.11 should be viewed as approximations of *expectations* of the corresponding signals in Fig. 8.9 (conditioned on the previous observations). Note that, for $\{+1, -1\}$ valued signals, the mean coincides with the difference $\tilde{p}(+1) - \tilde{p}(-1)$. It follows that the multiplier ③ in Fig. 8.11 computes (the continuous-time analog of) the message $\mu_{A,k}(t)$ according to (8.12); the box ④ in Fig. 8.11 computes (the continuous-time analog of) the message μ_k according to (8.13); and the box ⑤ computes (the continuous-time analog of) the message $\mu_{A,k}$ according to (8.11). All these computations can be done by simple transistor circuits as described in [121] [123] [116] (where the pseudo-probabilities $\tilde{p}(+1)$ and $\tilde{p}(-1)$ are represented by a pair of currents).

Consider next the filtered signals. Let $S_1(t)$ denote the output signal of the filter $H_1(s)$ in Fig. 8.9 and let $h_1(t)$ be the impulse response of that filter (i.e., the inverse Laplace transform of $H_1(s)$). We thus have

$$S_1(t) = \int_{-\infty}^{\infty} h_1(\tau)\, X(t-\tau)\, d\tau \tag{8.16}$$

and

$$\mathrm{E}\left[S_1(t)\right] = \int_{-\infty}^{\infty} h_1(\tau)\, \mathrm{E}\left[X(t-\tau)\right]\, d\tau \tag{8.17}$$

where the expectation is a (time dependent) ensemble average based on the (time dependent) pseudo-probabilities $\tilde{p}(+1)$ and $\tilde{p}(-1)$. It follows that the output of the filter $H_1(s)$ in Fig. 8.11—which is given by the right-hand side of (8.17)—is the expected value of $S_1(t)$. In other words, *all* signals in Fig. 8.11 may be viewed as (approximations of) expectations of the corresponding signals in Fig. 8.9.

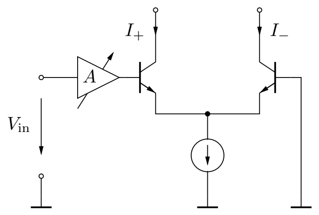


Figure 8.12: Differential transistor pair.

So far, all computations have been locally exact in the same sense as in the discrete-time case (i.e., ignoring cycles in the factor graph). This fails, however, for the threshold elements in Fig. 8.9: the (instantaneous) expectation of the output signal of such a threshold element is not determined by the (instantaneous) expectation of its input signal. At this point, however, practical considerations strongly suggest to implement the boxes ① and ② by the circuit of Fig. 8.12. This circuit accepts as input a voltage and produces as output two currents I_+ and I_- proportional to $\tilde{p}(+1)$ and $\tilde{p}(-1)$, respectively.

This same circuit is also used to implement the box ⑤ *exactly* (where the amplification A depends on the SNR and on the temperature). As an implementation of ① and ②, the circuit is an approximation; it would be exact (for the correct choice of α) if the distribution of the filtered signals —more precisely, the full sum-product message at the input of the soft-threshold elements—would be the logistic distribution

$$f(x) = \frac{1}{\beta\left(e^{\frac{x-\mu}{2\beta}} + e^{-\frac{x-\mu}{2\beta}}\right)^2} \tag{8.18}$$

with mean μ and variance $\pi\beta/\sqrt{3}$ [134, Appendix E]. In our experiments, the amplification α of these circuits was manually adjusted for the best performance.

8.7 Some Measurements

Simulation results of analog circuits are often subject to doubt concerning their robustness with respect to non-idealities. We therefore built the system of Fig. 8.11 as an actual (clockless) electronic circuit with discrete components. The filters were realized as active RC filters with integrated operational amplifiers.

For the measurements, the clean signal $X(t)$ as well as the noisy signal $Y(t)$ were created by *simulating* the circuit of Fig. 8.9 on a (digital) computer; the noisy signal $Y(t)$ was then passed as input to the electronic realization of Fig. 8.11. A typical measured output signal $\hat{X}(t)$ is shown in Fig. 8.10 (bottom).

Some measurements of this system are given in Figures 8.13–8.15. For the measurements of Figures 8.13 and 8.14, the signal $Y(t)$ is replaced by a constant signal with value -1 for $t < 0$. Both figures show the squared error (SE) $(\hat{X}(t) - X(t))^2$ averaged, first, over a sliding window and then, over a number of experiments. Fig. 8.13 shows the SE (averaged over 10ms and over 5 experiments) vs. the time t; Fig. 8.14 shows the SE (averaged over 1s and over 5 experiments) vs. the SNR at time $t = 4$ s (which is the steady state). Note that the receiver achieves good synchronization for an SNR down to about 0 dB. Not surprisingly, a signal with a longer period (top in Fig. 8.14) is more difficult to synchronize than a signal with a shorter period (bottom in Fig. 8.14).

It is instructive to observe what happens when the input to the receiving circuit is switched off for a while as illustrated in Fig. 8.15. Before the interruption, the receiver is synchronized. The signal $Y(t)$ is then masked (i.e., overwritten by zero) for 20 ms. During the interruption, $X(t)$ and $\hat{X}(t)$ drift apart and the averaged SE increases. The figure shows the signals $X(t)$ and $\hat{X}(t)$ around the critical moment when $Y(t)$ is switched on again.

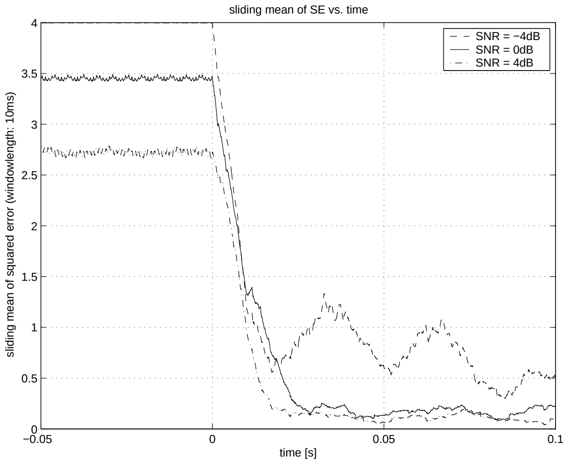


Figure 8.13: Average squared error vs. time after switching the transmission on.

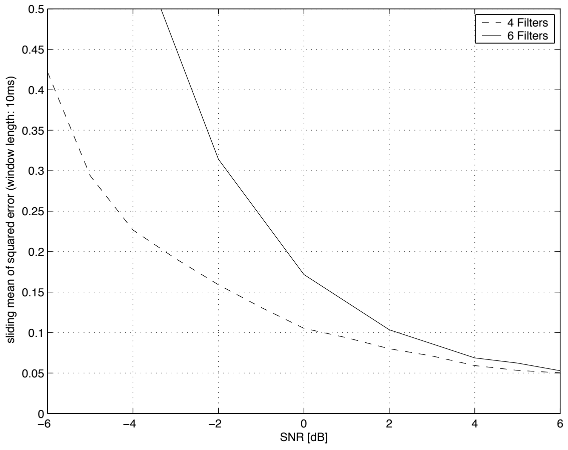


Figure 8.14: Average squared error in steady state vs. SNR. Dashed curve: pseudo-random signal with shorter period (7 ms instead of 34 ms, achieved with $H_2(s) = H_1(s)^4$ instead of $H_2(s) = H_1(s)^6$).

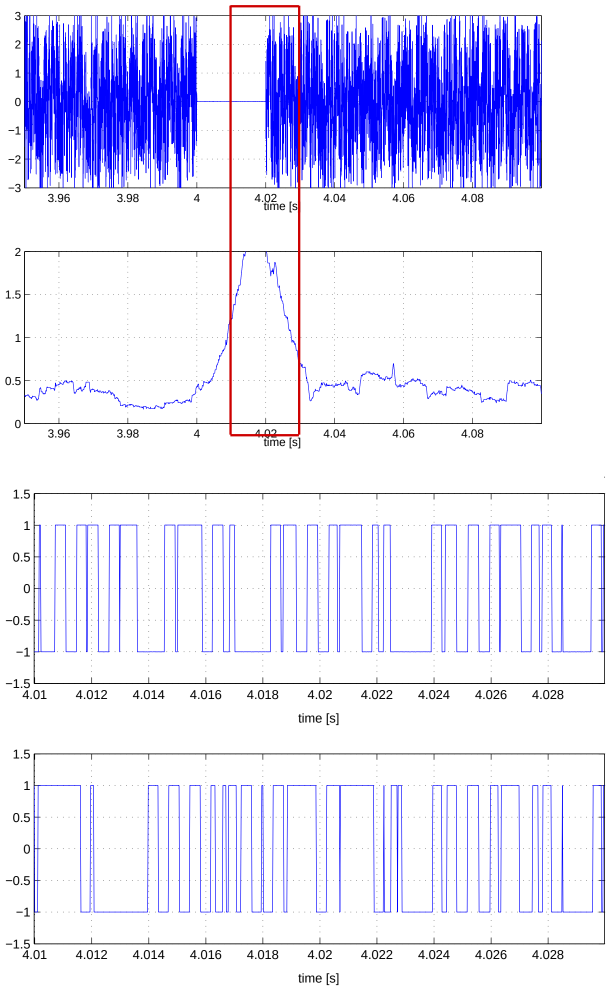


Figure 8.15: Resynchronization example with modified $Y(t)$ (top), sliding-window squared error (2nd from top), $X(t)$ (2nd from bottom), and $\hat{X}(t)$ (bottom) at SNR = 0 dB. The plots of $X(t)$ and $\hat{X}(t)$ are zoomed to the interval marked by the dashed lines.

8.8 Summary

- Gershenfeld and Grinstein demonstrated the synchronization of LFSR sequences (both in discrete time and in continuous time) by an "analog LFSR", which was obtained by embedding the discrete state space of the LFSR into a larger continuous state space. In this chapter, we derived such **dynamical systems** from **message-passing algorithms** for statistical state estimation.
- First, we noted that the soft LFSR proposed by Yang and Hanzo may be obtained by **forward-only message passing** through a factor graph.
- Second, we proposed a new **continuous-time analog** of both the LFSR and the soft LFSR that can be realized as a practical electronic circuit.
- We have thus established a **connection** between **statistical state estimation** and the phenomenon of **entrainment** of dynamical systems.

8.9 Outlook

- Dynamical systems (e.g., electronic circuits) with better entrainment capabilities may be obtained from **more powerful** (more complex) **message-passing algorithms**.
- So far, only message-passing algorithms for **detection** (i.e., inference of *discrete* variables) have been implemented in analog electronic circuits, i.e., analog *decoding* circuits (see [123] and references therein) and the *pseudo-noise synchronization* circuit we presented in this chapter. The extension to **estimation** seems to be largely unexplored. **Expectation-Maximization**-based and **gradient-based** estimation algorithms seem natural candidates for implementation in analog electronic circuits. EM often leads to simple expressions, whereas gradient-based algorithms naturally lead to feedback loops.

  One immediate application of this idea in the area of **communications** is **code-aided estimation** of the channel state, which may lead to **adaptive** analog circuits that are able to track the channel state. However, there are also potential applications beyond communications. In the late 80's, Mead [130] pioneered **neuromorphic engineering**, i.e., the development of artificial computing systems that use the physical properties and information representations found in biological nervous systems. Following that principle, Mead [130] (and later also other researchers, e.g., [113]) developed low-power analog circuits for applications in low-level vision ("silicon retina") and audio-signal processing ("silicon cochlea"). In such applications (e.g., in hearing aids), the available power is seriously limited, and digital solutions are often not suitable. It seems promising to underpin and extend the neuromorphic engineering paradigm with insights from **statistical estimation** (the "message-passing paradigm"), following the line of thought in this chapter.

- **Quantum systems** intrinsically compute probabilities. In fact, they do that in a very efficient manner: they can update the probability mass function of n bits in a *single* computation; on a classical computer, this requires in general in the order of 2^n computations. This fact may lead to efficient implementations of detection and estimation algorithms. Although we made some progress in this area, we are still far from a first working (toy) system. A major bottleneck is the fact that inference algorithms require **non-unitary operations** and **marginalization**; it is not clear how such operations can be implemented **efficiently** in quantum-computing systems.

Chapter 9

Conclusions and Outlook

9.1 Summary

This thesis was, on the one hand, about a particular problem, i.e., carrier-phase synchronization; on the other hand, it was about general methods, i.e., message-passing algorithms operating on factor graphs. We will summarize our results by following the latter thread.

We described how factor graphs can be used for statistical inference, i.e., detection and estimation. Statistical inference is accomplished by sending messages along the edges of the graph ("summary propagation" or "message passing"). Different algorithms are obtained by different message types or different message update schedules.

We described various standard estimation/detection algorithms in signal processing and machine learning as message passing on factor graphs:

- particle methods, e.g., Gibbs sampling, particle filtering, importance sampling, simulated annealing, Markov-Chain Monte-Carlo methods
- gradient-based methods, e.g., steepest ascent/descent, natural-gradient algorithms

- expectation maximization (EM) and extensions, e.g., Monte-Carlo EM, gradient EM, SAGE, etc.
- decision-based methods (e.g., iterative conditional modes)
- the back-propagation algorithm for training feed-forward neural networks.

We determined the local message update rules for each of the above algorithms. They may be used as building blocks for novel estimation and detection algorithms. By listing the possible update rules at each node in the factor graph, one can *systematically* derive novel algorithms. We derived various phase-estimation algorithms in this fashion. We demonstrated how message-passing algorithms for inference can be implemented as dynamical systems, in particular, as analog electrical circuits. In particular, we have developed a clockless low-power analog circuit that synchronizes to pseudo-noise sequences.

Factor graphs can also be used for other (related) tasks. In this dissertation, we have devised message-passing algorithms to compute information matrices and Cramér-Rao-type bounds. The latter allow us to assess practical estimation algorithms, e.g., our phase-estimation algorithms. Our algorithms for computing information matrices may lead to novel natural-gradient-based algorithms.

Information matrices are also the key to kernel methods, i.e., Fisher kernels [89]. Our methods for computing information matrices may enable us to derive Fisher kernels from sophisticated graphical models. We also explained how probabilistic kernels and product kernels may be derived from graphical models. In combination with kernel methods, factor graphs can be used for virtually any task in machine learning, as for example clustering, classification, and novelty detection.

A different application of factor graphs is the computation of information rates of (discrete) communications channels with memory. In [12], it has been shown how information rates for such channels can be computed by forward-only messaging on the graph of the state-space model that represents the channel. We extended this result to continuous channels, e.g., channels with phase noise. We also described how the capacity (or lower bounds on the capacity) of continuous channels can be computed.

9.2 Outlook

Phase Estimation

- *Extension to other phase models.* We derived phase estimation algorithms for the constant-phase model and the random-walk phase model. Those algorithms could be extended to more sophisticated phase models, in particular, the models we described in this thesis.
- *Other synchronization tasks.* We merely focussed on phase estimation. Following the line of though of this thesis, message-passing algorithms for other synchronization tasks may be derived.
- *Analysis of synchronization algorithms.* We have proposed various algorithms for phase estimation and performed simulations to asses their performance. Our algorithms (and synchronization algorithms in general) may also be analyzed by semi-analytical tools as for example density evolution.

Computation of information rates and capacities

- *Information rates.* We proposed a method for computing the information rate of continuous channels with memory; as an illustration, we applied the method to the random-walk phase model. The techniques could also be applied to more sophisticated phase model and other types of channel models, e.g., related to timing synchronization.
- *Capacities of continuous channels with memory.* We outlined how our method for computing capacities (or lower bounds on capacities) for memoryless channels can be extended to channels with memory. We have recently implemented such algorithms, but they need to be further analyzed.

Novel applications

- *Kernel methods.* We outlined several general strategies to derive kernels from factor graphs, i.e., Fisher kernels and probabilistic kernels. In the machine learning literature, such kernels are usually computed on cycle-free factor graphs. The extension to cyclic factor graphs may lead to novel applications.

- *Information geometry.* Our methods to compute Fisher information matrices can be used for several other applications, i.e., for deriving Fisher kernels and natural-gradient-based algorithms for sophisticated graphical models. Such algorithms need to be implemented and tested.

- *Reinforcement Learning.* One of the learning methods which we did not address in this thesis is reinforcement learning. It needs to be investigated whether also this method could be integrated in the factor-graph framework.

- *Combining learning methods with model-based approaches.* In some applications, the probability function of only a *part* of the system may be known, for the other part of the system, only a list of i.i.d. samples may be given. Such systems may be represented by factor graphs in which some nodes are known, others are unknown and need to be learned from the available samples. The unknown nodes could be represented by a neural network, kernel machine or density tree. Such hybrid systems seem to be largely unexplored.

- *Message passing and dynamical systems.* We demonstrated how message-passing algorithms can be implemented in analog electronic circuits. Other natural candidates for the implementation of message-passing algorithms are opto-electronic systems and quantum systems.

- *Representation of signals.* In this thesis, all signals were represented in time-domain. In some applications, (a part of) the system is most naturally described in the Fourier domain. Alternatively, signals may be represented by means of wavelets or filterbanks. Related topics are *redundant* representations of signals (and systems) and (adaptive) quantization of continuous signals. Signal representation is a central theme in signal processing, which could be more intensively explored in the context of message passing.

Appendix A

Estimation and Decision Theory

In this Appendix, we review some basic notions from estimation, detection and learning theory. We will keep the exposition informal; we refer to [176] for a more rigorous and more detailed treatment.

A.1 Estimation theory

The standard estimation problem is depicted in Fig. A.1. Based on the measurement $Y = y$ of a (discrete or continuous) random variable Y with alphabet $\mathcal{Y}$, we wish to estimate the value x of a variable X. We assume that for each observation $Y = y$, a function $f_y(x)$ is at our disposal that encodes all available information about X. The "cost" associated with each estimate $\hat{x}$ of the true value x is quantified by a real-valued

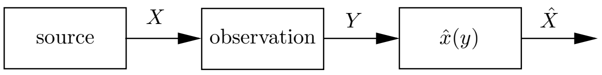


Figure A.1: Estimation problem.

function $\kappa(x, \hat{x})$. Note that the variables X and Y can be scalars, vectors, or matrices. In the following, we deal with the questions:

- *What is a "good" estimator?*
- *Which estimators are used in practice?*
  - *How do they generate an estimate $\hat{x}$ from $f_y(x)$ and $\kappa(x, \hat{x})$?*
  - *In what sense are they good?*

Obviously, there are various ways to define "good" estimators; one may be interested in the performance of the estimator

a) for *all* possible values x,

b) on the *average*,

c) in the *worst case.*

Minimax estimation tries to minimize the worst-case cost. We do not treat this topic here since it is not directly relevant for this thesis; we refer to [176, pp. 167–174] for more information.

In classical estimation theory, one focusses on the cost for *all* values of x simultaneously. The variables X and Y are in this context real or complex scalars or vectors and the cost function of interest is $\kappa(x, \hat{x}) \triangleq |x - \hat{x}|^2$. The variable X is regarded as a "non-random" parameter: one does not introduce a prior $p_X(x)$ for X. The function $f_y(x)$ is then a probability function in y *parameterized* by the unknown parameter x; the standard notation is $f_y(x) \triangleq p(y; x)$, but we will write $f_y(x) \triangleq p_{Y|X}(y|x)$ instead. We now investigate the key concepts in classical estimation theory: consistency, unbiasedness, and efficiency.

- **(Consistency)**
  Let y_1^n be a sequence of n observations, i.e., $y_1^n \triangleq (y_1, y_2, \ldots, y_n)$. We say that an estimator $\hat{x}\left(y_1^n\right)$ that estimates the value of X based on the sequence y_1^n is **consistent** if

$$\lim_{n\to\infty} \hat{x}\left(Y_1^n\right) = X \qquad \text{w.p.1.} \tag{A.1}$$

- **(Unbiasedness)**
  An estimator $\hat{x}(y)$ is called **unbiased** if
  $$\mathrm{E}[\hat{x}(Y)|X=x] \triangleq \int_y \hat{x}(y) p_{Y|X}(y|x) dy = x \tag{A.2}$$
  for all x.

- **(Efficiency)**
  Assume X is real or complex. If an estimator minimizes the estimation error $\mathrm{MSE}(x,\hat{x}) \triangleq \mathrm{E}[|\hat{x}(Y)-x|^2|X=x]$, it is said to be **efficient**. If it minimizes $\mathrm{E}[|\hat{x}(Y)-x|^2|X=x]$ for all x, it is called **uniformly efficient**.

Remark A.1. (On consistency)
It seems reasonable to require that an estimator attains the true value in the limit of an infinite amount of data. Most estimators used in practice are consistent.

Remark A.2. (On unbiasedness)
For particular estimation problems, all estimators are necessarily *biased.* For example, assume that X takes values in a finite interval $[a,b]$ or an half-infinite interval $[a,\infty)$. If
$$\min_{\hat{x}(y)} \mathrm{E}[|\hat{x}(Y)-a|^2|X=a] > 0, \tag{A.3}$$
all estimators $\hat{x}(y)$ are biased.

Remark A.3. (On efficiency)

- At first sight, efficiency may seem a natural candidate for an optimality criterion. Unfortunately, this criterion can not directly be adopted: minimizing the error function $\mathrm{E}[|\hat{x}(Y)-x|^2|X=x]$ w.r.t. $\hat{x}(y)$ leads to the trivial solution: $\hat{x}(y)=x$.[1] Such an estimator is obviously not realizable, since x is the unknown value we are trying to find out! This problem may in fact occur for *any* criterion of the form $\mathrm{E}[\kappa(x,\hat{x}(Y))|X=x]$.

- The classical approach to solve this problem, is to focus on *unbiased* estimators with minimum $\mathrm{MSE}(x,\hat{x})$ (for all x). For some estimation problems, such estimators exist and can be implemented.

[1] This can be proved by solving the corresponding Euler-Lagrange differential equation.

However, a *biased* estimator may have a smaller MSE$(x, \hat{x})$ than any unbiased estimator (see [186] for a convincing example). Even worse, the set of unbiased estimators may be empty, as we stated before.

In many practical situations, an unbiased estimator with minimal MSE$(x, \hat{x})$ for all x cannot be found or does not exist; a popular alternative is the maximum likelihood estimator.

Maximum likelihood estimation:

$$\hat{x}^{\mathrm{ML}}(y) = \operatorname*{argmax}_{x \in \mathcal{X}} p_{Y|X}(y|x).$$

The maximum likelihood estimate $\hat{x}_{\mathrm{ML}}$ is the *argmax* (a.k.a "mode") of the function $f_y(x) \triangleq p_{Y|X}(y|x)$.

In Bayesian estimation, one tries to find an estimator that minimizes the *expected* cost, which is the cost *averaged* over all values of x of X:

$$\begin{aligned} \mathrm{E}[\kappa(X, \hat{x})] &\triangleq \mathrm{E}_Y\big[\mathrm{E}_{X|Y}[\kappa\big(X, \hat{x}\big)|Y = y]\big], && \text{(A.4)} \\ &\triangleq \int_x \int_y \kappa(x, \hat{x}(y)) p_{Y|X}(y|x) p_X(x) dx dy. && \text{(A.5)} \end{aligned}$$

In other words, instead of trying to find an estimator that has the best performance for *all* values x simultaneously—which is asking too much, as we have seen—one hopes to find an estimator that has the best performance on the average. Obviously, for a *given* value x, there may be an estimator that yields a smaller cost than the Bayesian estimator.

In this context, the variable X is treated as a random variable with prior $p_X(x)$. The function $f_y(x)$ is defined as

$$\begin{aligned} f_y(x) &\triangleq p_{X|Y}(x|y) && \text{(A.6)} \\ &= \frac{p_{Y|X}(y|x) p_X(x)}{\int_x p_{Y|X}(y|x) p_X(x) dx}, && \text{(A.7)} \end{aligned}$$

where the equality (A.7) is known as Bayes' rule—hence the name "Bayesian estimation".

How does a Bayesian estimator generate an estimate $\hat{x}$ from $f_y(x) \triangleq p_{X|Y}(x|y)$? If one minimizes the conditional cost for any observation $Y = y$, i.e.,

$$\mathrm{E}[\kappa(X,\hat{x})|Y=y] \triangleq \int_x \kappa(x,\hat{x}(y))p_{X|Y}(x|y)dx, \tag{A.8}$$

then one obtains the Bayesian estimation rule:[2]

Bayesian estimation:

$$\hat{x}^{\mathrm{BAYES}}(y) = \operatorname*{argmin}_{\hat{x}\in\mathcal{X}} \mathrm{E}[\kappa(X,\hat{x})|Y=y].$$

The Bayesian estimator also minimizes the average cost (A.4). When X and $\hat{X}$ are real or complex valued and $\kappa(x,\hat{x}) = |\hat{x}-x|^2$, then the Bayesian estimator reduces to the minimum mean squared error estimator (MMSE).

Minimum mean squared error estimation:

$$\hat{x}^{\mathrm{MMSE}}(y) \triangleq \mathrm{E}[X|Y=y].$$

The MMSE estimator is thus given by the expectation of the conditional probability function $f_y(x) \triangleq p_{X|Y}(x|y)$ and yields the estimation error

$$\mathrm{E}[|X-\hat{x}^{\mathrm{MMSE}}(y)|^2|Y=y] = \mathrm{Var}[X|Y=y]. \tag{A.9}$$

The conditional expectation is in fact the optimal estimate for a large class of cost functions, as the following theorem indicates [199, pp. 60–62].

Theorem A.1. If the cost function $\kappa(x,\hat{x})$

- only depends on the error $\tilde{x} \triangleq x - \hat{x}$, i.e., $\kappa(x,\hat{x}) \triangleq \kappa(\tilde{x})$,
- is symmetric, i.e., $\kappa(\tilde{x}) = \kappa(-\tilde{x})$ for all $\tilde{x}$,
- is convex, i.e., $\kappa(ax_1 + (1-a)x_2) \geq a\kappa(x_1) + (1-a)\kappa(x_2)$ for any $a \in (0,1)$ and for all x_1, x_2,

[2]Strictly speaking, argmin returns a set. If the set contains several elements, then we randomly pick one of these; otherwise, we return the single element.

and the posterior density $p_{X|Y}(x|y)$ is symmetric about its mean $m_X \triangleq \mathrm{E}[X|Y = y]$, i.e., $p_{X|Y}(m_X - x|y) = p_{X|Y}(m_X + x|y)$ for all x, then the estimate $\hat{x}$ that minimizes any average cost $\mathrm{E}[\kappa(\tilde{X})]$ in this class is identical to

$$\hat{x}^{\mathrm{MMSE}} \triangleq m_X \triangleq \mathrm{E}[X|Y = y]. \tag{A.10}$$

This invariance to the choice of a cost function is a useful feature, since it is often not a priori clear which $\kappa(\tilde{x})$ should be chosen. In many estimation problems, however, the density $p_{X|Y}(x|y)$ is not symmetric about its mean and, as a consequence, Theorem A.1 does not hold.

Besides Bayesian estimators, also widely used in practice is the maximum a-posteriori estimator (MAP).

Maximum a-posteriori estimation:

$$\hat{x}^{\mathrm{MAP}}(y) = \operatorname*{argmax}_{x \in \mathcal{X}} p_{X|Y}(x|y).$$

Summarizing:

Practical estimators differ in the way the function $f_y(x)$ is determined, and how the estimate $\hat{x}$ is extracted from $f_y(x)$:

- in Bayesian and MAP estimation, the **function** $f_y(x)$ is given by the posterior probability function $p_{X|Y}(x|y)$; in ML estimation, $f_y(x) \triangleq p_{Y|X}(y|x)$.
- in ML and MAP estimation, the **estimate** $\hat{x}$ is the *argmax* (a.k.a. "mode") of $f_y(x)$; in the MMSE estimator, it is the *mean.*

Remark A.4. (Likelihood)
The function $p_{Y|X}(y|x)$, viewed as a function of x (with fixed value of y), is often called the "likelihood function", hence the name maximum likelihood estimation.

Remark A.5. (Blockwise vs. symbolwise estimation)
Assume $\mathbf{X}$ and $\mathbf{Y}$ are random vectors. The blockwise maximum a-posteriori estimator is given by

$$\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} p_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}), \tag{A.11}$$

whereas the symbolwise MAP-estimator equals

$$\hat{x}_i^{\mathrm{MAP}}(\mathbf{y}) = \underset{x_i}{\operatorname{argmax}}\, p_{X_i|\mathbf{Y}}(x_i|\mathbf{y}). \tag{A.12}$$

Note that the vector of symbolwise MAP-estimates $(\hat{x}_1^{\mathrm{MAP}}, \hat{x}_2^{\mathrm{MAP}}, \ldots, \hat{x}_n^{\mathrm{MAP}})$ is in general not equal to the blockwise MAP-estimate $\hat{\boldsymbol{x}}^{\mathrm{MAP}}$. Similarly, the blockwise and symbolwise maximum likelihood estimators are defined as

$$\hat{\boldsymbol{x}}^{\mathrm{ML}}(\mathbf{y}) = \underset{\mathbf{x}}{\operatorname{argmax}}\, p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}), \tag{A.13}$$

and

$$\hat{x}_i^{\mathrm{ML}}(\mathbf{y}) = \underset{x_i}{\operatorname{argmax}}\, p_{\mathbf{Y}_i|X_i}(\mathbf{y}|x_i). \tag{A.14}$$

respectively, where generally $(\hat{x}_1^{\mathrm{ML}}, \hat{x}_2^{\mathrm{ML}}, \ldots, \hat{x}_n^{\mathrm{ML}}) \neq \hat{\boldsymbol{x}}^{\mathrm{ML}}$. Along the same lines, one can define the blockwise and symbolwise MMSE estimator. If $\hat{x}_i^{\mathrm{MMSE}}$ is the symbolwise MMSE estimate of x_i, then $(\hat{x}_1^{\mathrm{MMSE}}, \hat{x}_2^{\mathrm{MMSE}}, \ldots, \hat{x}_n^{\mathrm{MMSE}}) = \hat{\boldsymbol{x}}^{\mathrm{MMSE}}$, where $\hat{\boldsymbol{x}}^{\mathrm{MMSE}}$ is the blockwise estimate.

Remark A.6. (Classical learning)
In the estimation problem, a function $f_y(x)$ is given that encodes all available information about the true value x. If we instead only have a finite number of i.i.d. samples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ from the probability function $p_{XY}(x, y)$ at our disposal, then the problem is not called estimation but "learning". Many learning algorithms are based (explicitly or implicitly) on a proposal function $\hat{p}_{XY|\Theta}(x, y|\theta)$ (parameterized by θ) that "models" the true (unknown) function $p_{XY}(x, y)$. In the classical learning paradigm, the parameters θ are *estimated* from the available data $\mathcal{D}$, for example by the ML rule

$$\hat{\theta}^{\mathrm{ML}} = \underset{\theta}{\operatorname{argmax}}\, p(\mathcal{D}|\theta), \tag{A.15}$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p_{XY|\Theta}(x_i, y_i|\theta), \tag{A.16}$$

or approximations thereof. The model $\hat{p}_{XY}(x, y)$ is given by

$$\hat{p}_{XY}(x, y) \stackrel{\triangle}{=} \hat{p}_{XY|\Theta}(x, y|\hat{\theta}). \tag{A.17}$$

An estimate $\hat{x}$ of the true value x may for example be obtained by the MAP rule based on $\hat{p}_{XY|\Theta}$.

In this setting, learning boils down to estimation. The boundary between learning and estimation is generally rather fuzzy, and the estimation techniques we propose in this thesis can often be applied to problems in machine learning.

Remark A.7. (Bayesian learning)
In the Bayesian learning paradigm, the learning parameters θ are interpreted as random variables with prior $p_\Theta(\theta)$. One usually chooses a prior $p_\Theta(\theta)$ that assigns most probability to "simple" models. The parameters θ are not *estimated* explicitly; instead, the model $\hat{p}_{XY}(x, y)$ is obtained by *marginalizing* over θ

$$\hat{p}_{XY}(x, y) \triangleq \int_\theta p_{XY|\Theta}(x, y|\theta) p_\Theta(\theta) d\theta. \tag{A.18}$$

One advantage of this approach is that the uncertainty concerning θ is explicitly taken into account. In contrast, the model (A.17) does not incorporate this information, it merely uses the estimate $\hat{\theta}$. In addition, the Bayesian paradigm automatically amounts to simple models that sufficiently explain the data without unnecessary complexity. We refer the reader to the excellent tutorial [193] for additional information on Bayesian methods in machine learning.

A.2 Decision theory

We still consider the setup depicted in Fig. A.1, where we wish to infer the value of X based on an observation $Y = y$. In contrast to the previous section, we assume here that X is a *discrete* random variable, i.e., a stochastic variable that takes value in a finite or countable infinite set $\mathcal{X}$. The problem is then not called *estimation*, but *detection* instead.

The Bayesian detection rule is given as follows.

Bayesian detection:

$$\begin{aligned}\hat{x}^{\text{BAYES}}(y) &= \operatorname*{argmin}_{\hat{x} \in \mathcal{X}} \mathrm{E}[\kappa(X, \hat{x})|Y = y] \\ &= \operatorname*{argmin}_{\hat{x} \in \mathcal{X}} \sum_{x \in \mathcal{X}} \kappa(x, \hat{x}) p_X(x) p_{Y|X}(y|x).\end{aligned}$$

Of special interest is the error function

$$\kappa(x, \hat{x}) = \begin{cases} 0, & \hat{x} = x \\ 1, & \hat{x} \neq x. \end{cases} \tag{A.19}$$

The conditional expected cost is then equal to the probability of error given the observation $Y = y$

$$\begin{aligned} \mathrm{E}[\kappa(X, \hat{x})|Y = y] &= \sum_{x \in \mathcal{X} : \hat{x} \neq x} P(X = x|Y = y) && \text{(A.20)} \\ &= P(\hat{x} \neq x|Y = y). && \text{(A.21)} \end{aligned}$$

The Bayesian detection rule that minimizes this error probability is called the maximum a-posteriori detector.

Maximum a-posteriori detection:

$$\begin{aligned} \hat{x}^{\mathrm{MAP}}(y) &= \operatorname*{argmin}_{x \in \mathcal{X}} P(\hat{x} \neq x|Y = y) \\ &= \operatorname*{argmax}_{x \in \mathcal{X}} P_{X|Y}(x|y). \end{aligned}$$

The maximum likelihood rule remains unchanged.

Maximum likelihood detection:

$$\hat{x}^{\mathrm{ML}}(y) = \operatorname*{argmax}_{x \in \mathcal{X}} P_{Y|X}(y|x).$$

We now discuss the maximum a-posteriori decision rule for several scenarios important in (channel) decoding.

Example A.1. (Block-wise decoding)
We set

$$\begin{aligned} X &= \mathbf{U} && \text{(vector to be estimated)} \\ Y &= \mathbf{Y} && \text{(measurement)} \\ \hat{X} &= \hat{\mathbf{U}}^{\text{block}} && \text{(estimate).} \end{aligned}$$

Minimization of $P_{\text{error}} = P_{\text{block}} = P[\hat{\mathbf{U}} \neq \mathbf{U}]$ leads to the decision rule

$$\boxed{\hat{\mathbf{u}}^{\text{block}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{UY}}(\mathbf{u},\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} \max_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y}).}$$

The introduction of the codeword vector $\mathbf{X}$ does not change the problem as $\mathbf{X}$ is assumed to be a deterministic function of $\mathbf{U}$ (see Section 2.1). The decision taken for component i $(1 \le i \le k)$ of $\hat{\mathbf{u}}^{\text{block}}(\mathbf{y})$ can also be written as

$$\boxed{\hat{u}_i^{\text{block}}(\mathbf{y}) = \operatorname*{argmax}_{u_i\in\mathcal{U}} \max_{\substack{\mathbf{u}\in\mathcal{U}^k \\ u_i \text{ fixed}}} P_{\mathbf{UY}}(\mathbf{u},\mathbf{y}) = \operatorname*{argmax}_{u_i\in\mathcal{U}} \max_{\substack{\mathbf{u}\in\mathcal{U}^k,\mathbf{x}\in\mathcal{X}^n \\ u_i \text{ fixed}}} P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y}).}$$

Note: The well-known Viterbi-Algorithm efficiently performs the task of finding the block-wise estimate in the case of convolutional or trellis codes. □

Example A.2. (Symbol-wise decoding)
For each $i = 1, \dots, k$ we set

$$\begin{aligned} X &= U_i && \text{(variable to be estimated)} \\ Y &= \mathbf{Y} && \text{(measurement)} \\ \hat{X} &= \hat{U}_i^{\text{symbol}} && \text{(estimate).} \end{aligned}$$

Minimization of $P_{\text{error}} = P_{\text{symbol}}^{(i)} = P[\hat{U}_i \neq U_i]$ leads to the decision rule

$$\boxed{\begin{aligned} \hat{u}_i^{\text{symbol}}(\mathbf{y}) &= \operatorname*{argmax}_{u_i\in\mathcal{U}} P_{U_i\mathbf{Y}}(u_i,\mathbf{y}) \\ &= \operatorname*{argmax}_{u_i\in\mathcal{U}} \sum_{\substack{\mathbf{u}\in\mathcal{U}^k,\ \mathbf{x}\in\mathcal{X}^n \\ u_i \text{ fixed}}} P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y}) \quad (i = 1, \dots, k). \end{aligned}}$$

□

Remark A.8. (Marginalization and maximization)
The joint probability function $P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y})$ appears in the expressions

for block-wise- and symbol-wise-decoding. However, one needs to perform different operations to get the block-wise/symbol-wise estimates; *marginalization* leads to the symbol-wise estimates, whereas the block-wise estimates are obtained by *maximization*. Both operations are at the heart of many detection and estimation algorithms; in Chapter 3, we show that those operations can be carried out as message passing on cycle-free factor graphs. In the case of *cyclic* factor graphs, one can still apply the same message-passing algorithm, but the results might not be the same as with "correct" block- or symbol-wise decoding as defined in Examples A.1 and A.2.

Appendix B

Notions from Information Theory

Let us consider Fig. 2.1, which depicts a basic block diagram of a digital communications systems. In Section 2.1, we shortly describe each individual block; here, we address a question that concerns *the system as a whole*:

> *What is the highest rate R at which information can be transmitted over a given physical channel?*

Shannon [178] has proved that it is possible to transmit data with *as few errors as desired* if one is willing to use (very long) channel codes of rates smaller than *channel capacity*; we refer to this operation mode as *reliable communications.* In other words, the number of "non-confusable" waveforms for n uses of a communications channel grows exponentially with n, and the exponent is the channel capacity. Shannon established that the capacity of a wide variety of channels can be expressed as the maximum mutual information between the channel input and output [178]; in other words, Shannon's channel capacity theorem not only promises that reliable communication is possible at a *non-zero* rate, it also provides an explicit expression for the maximum rate. This expression is, however, often intractable. Moreover, Shannon's channel capacity theorem gives us almost no practical guidelines on how optimal transmission can be achieved.

In the following, we will review the channel capacity theorem for memoryless channels and for stationary ergodic channels with memory. But first, we need to introduce some important concepts; in this section, we closely follow the book of Cover and Thomas [37].

B.1 Definitions and theorems

We introduce the concept of entropy, which is a measure for the uncertainty of a random variable.

Definition B.1. (Entropy of a discrete random variable)
The entropy of a discrete random variable X with probability mass function (pmf) $p_X(x)$ is defined as

$$H(X) \triangleq -\sum_x p_X(x) \log_b p_X(x). \tag{B.1}$$

$\square$

The entropy $H(X)$ is a functional of the distribution $p_X(x)$, it does not depend on the actual values taken by the random variable X. Moreover, the entropy $H(X)$ is concave[1] in $p_X(x)$. The choice of the base b determines the unit. When $b = 2$, the unit is called *bit*. When $b = e$, the other base commonly used in information theory, the unit is called *nat*. In the sequel, we will use the logarithm to the base $b = 2$ and the unit is thus *bit*. We will use the convention that $0 \log_2 0 = 0$.

We extend the previous definition to pairs of random variables.

Definition B.2. (Joint entropy)
The joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with joint distribution $p(x, y)$ is given by

$$H(X,Y) \triangleq -\sum_{x,y} p_{XY}(x,y) \log_2 p_{XY}(x,y). \tag{B.2}$$

[1] A *function* f is called *concave*, if $-f$ is convex; a function f is said to be *convex* if for all x_1, x_2 and $0 \le \lambda \le 1$, $f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda) f(x_2)$. Convex and concave *functionals* are defined likewise.

□

We define the conditional entropy of a random variable given another.

Definition B.3. (Conditional entropy)
The entropy of a discrete random variable X conditioned on a discrete variable Y is given by

$$H(X|Y) \triangleq -\sum_{x,y} p_{XY}(x,y)\log_2 p_{X|Y}(x|y). \tag{B.3}$$

□

Differential entropies, joint and conditional differential entropies of continuous random variables are defined by replacing the summation by integration.[2] They are denoted by the lower-case letter h, i.e., $h(X)$, $h\big(X|Y\big)$, and $h\big(X,Y\big)$ respectively. In the rest of this section, we make the following conventions: (1) if x is discrete, $\sum_x g(x)$ stands for the summation of $g(x)$ over its support, otherwise, it stands for integration; (2) the expression $H(\cdot)$ stands for entropy if the argument of $H(\cdot)$ is discrete, it stands for differential entropy otherwise.

The notions of conditional entropy and joint entropy are connected by a chain rule.

Theorem B.1. (Chain rule)

$$H(X,Y) = H(X) + H(Y|X). \tag{B.4}$$

The divergence is a measure of the "distance" between two probability distributions p and q.

Definition B.4. (Divergence)
The divergence between two probability distributions $p(\cdot)$ and $q(\cdot)$ is defined as

$$D\big(p\|q\big) \triangleq \sum_{x} p(x)\log_2 \frac{p(x)}{q(x)}. \tag{B.5}$$

□

[2]As in every definition involving an integral or a density, we should include the statement *if it exists*.

In the above definition, we use the convention that $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = \infty$. The divergence is always positive with equality if and only if $p = q$. It is not a true distance, as it is not symmetric, i.e., in general $D\big(p\|q\big) \neq D\big(q\|p\big)$. The divergence is also called “relative entropy” or “Kullback-Leibler distance”.

Theorem B.2. (Convexity of divergence)
The divergence $D\big(p\|q\big)$ is convex in the pair (p, q), i.e., if (p_1, q_1) and (p_2, q_2) are two pairs of probability mass functions, then

$$D\big(\lambda p_1 + (1-\lambda)p_2 \| \lambda q_1 + (1-\lambda)q_2\big) \leq \lambda D\big(p_1\|q_1\big) + (1-\lambda)D(p_2\|q_2) \quad \text{(B.6)}$$

for all $0 \leq \lambda \leq 1$.

We now introduce mutual information, which is a measure of the amount of information that one random variable contains about another random variable; alternatively, it may be interpreted as the reduction in uncertainty of one random variable due to the knowledge of the other.

Definition B.5. (Mutual information)
Consider two random variables X and Y with joint probability mass function $p(x, y)$. The mutual information $I(X; Y)$ is the relative entropy between the joint probability mass function $p_{XY}(x, y)$ and the product $p_X(x)p_Y(y)$ distribution, i.e.,

$$\begin{aligned} I(X;Y) &\triangleq \sum_{x,y} p_{XY}(x,y) \log_2 \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)} && \text{(B.7)} \\ &= \sum_{x,y} p_X(x)p_{Y|X}(y|x) \log_2 \frac{p_{Y|X}(y|x)}{p_Y(y)}, && \text{(B.8)} \end{aligned}$$

where $p_Y(y)$ is the output distribution defined as

$$p_Y(y) \triangleq \sum_x p_X(x)p_{Y|X}(y|x). \quad \text{(B.9)}$$

□

Remark B.1. (Mutual information and entropies)
In terms of entropies, we can write the mutual information as

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) && \text{(B.10)} \\ &= H(Y) - H(Y|X). && \text{(B.11)} \end{aligned}$$

Remark B.2. (Mutual information as divergence)
The mutual information between the input X and the output Y of a discrete memoryless channel can be rewritten as a divergence in the following way

$$I(X;Y) = \sum_x p_X(x) \sum_y p_{Y|X}(y|x) \log_2 \frac{p_{Y|X}(y|x)}{p_Y(y)} \tag{B.12}$$

$$= \sum_x p_X(x) D\big(p_{Y|X}(\cdot|x) \| p_Y(\cdot)\big). \tag{B.13}$$

Theorem B.3. (Convexity of mutual information)
The mutual information $I(X;Y)$ is a concave function of $p_X(x)$ for fixed $p_{Y|X}(y|x)$ and a convex function of $p_{Y|X}(y|x)$ for fixed $p_X(x)$.

We now consider stochastic processes. The entropy rate of a stochastic process is the rate at which the entropy grows with n.

Definition B.6. (Entropy rate)
The entropy rate of a stochastic process X is defined as

$$H(X) \triangleq \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n), \tag{B.14}$$

when the limit exists. The RHS expression is the per-symbol entropy rate. One can also define a related quantity for entropy rate:

$$H'(X) \triangleq \lim_{n\to\infty} \frac{1}{n} H(X_n | X_{n-1}, \ldots, X_1), \tag{B.15}$$

where the RHS expression is the conditional entropy rate of the last random variable given the past. For stationary stochastic processes, both quantities $H(X)$ and $H'(X)$ are equal. □

Let us now consider two important examples.

Example B.1. (i.i.d. Process)
Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed (i.i.d.) random variables. Applying the chain rule yields

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) = H(X_k). \tag{B.16}$$

□

Example B.2. (Markov chain)
As a consequence of the Markov property, the entropy rate of a Markov Chain is

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_n|X_{n-1},\ldots,X_1) = H(X_n|X_{n-1}). \tag{B.17}$$

□

Definition B.7. (Information rate)
The information rate between two stationary and ergodic processes X and Y is defined as

$$\begin{aligned} I(X;Y) &\triangleq \lim_{n\to\infty} \frac{1}{n} I(X_1,X_2,\ldots,X_n;Y_1,Y_2,\ldots,Y_n) && \text{(B.18)}\\ &= \lim_{n\to\infty} \frac{1}{n}\Big[H(X_1,X_2,\ldots,X_n) \\ &\qquad - H(X_1,X_2,\ldots,X_n|Y_1,Y_2,\ldots,Y_n)\Big] && \text{(B.19)}\\ &= H(X) - H(X|Y) && \text{(B.20)}\\ &= H(Y) - H(Y|X) && \text{(B.21)} \end{aligned}$$

when the limit in (B.18) exists. □

If X is the input and Y is the output process of a communications channel, the limit in (B.18) exists if the channel law preserves the property of stationarity and ergodicity of the input process. We call such channels ergodic channels.

Of crucial importance in information theory is the **Asymptotic Equipartition Property** (AEP), which is related to the notion of **typicality**; AEP is the key to data compression and is one of the main ingredients in Shannon's proof of the channel-capacity theorem. It also opens the door to numerical algorithms for computing information rates and channel capacities, one of the topics of this thesis. AEP and typicality are a direct consequence of the weak law of large numbers.

Theorem B.4. (Weak law of large numbers)
Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables. Define the random variable $\bar{X}_n$ as

$$\bar{X}_n \triangleq \frac{1}{n}\sum_{k=1}^{n} X_k. \tag{B.22}$$

Then

$$\lim_{n\to\infty} \Pr[|\bar{X}_n - \mathrm{E}[X]| \geq \varepsilon] = 0, \quad \forall \varepsilon > 0. \tag{B.23}$$

Theorem B.5. (AEP for i.i.d. processes)
Let $X_1, X_2, \dots, X_n$ be i.i.d. random variables and distributed according to probability function $p_X(x)$. Then

$$\lim_{n\to\infty} \Pr\Big[\Big| -\frac{1}{n}\log p_X(X_1, X_2, \dots, X_n) - H(X)\Big| \geq \varepsilon\Big] = 0, \quad \forall \varepsilon > 0. \tag{B.24}$$

AEP allows us to divide the set of all sequences into two sets, the set of *typical* sequences, where the sample entropy is "close" to the ensemble entropy, and the set of *non-typical* sequences containing all other sequences. The following definition formalizes this idea.

Definition B.8. (Typical sequences)
Let $X_1, X_2, \dots, X_n$ be i.i.d. random variables and distributed according to probability function $p_X(x)$. A typical set $\mathcal{X}_\varepsilon^n$ with respect to the probability measure $p_X(x)$ is the set of sequences $(x_1, x_2, \dots, x_n) \in \mathcal{X}^n$ having the following property:

$$2^{-n(H(X)+\varepsilon)} \leq p_X(x_1, x_2, \dots, x_n) \leq 2^{-n(H(X)-\varepsilon)}. \tag{B.25}$$

□

As a consequence of the asymptotic equipartition property (AEP), the typical set has probability nearly 1 and all elements of the typical set are nearly equiprobable. The elements of the set are called *typical sequences* and their number is nearly $2^{nH(X)}$. All other sequences have probability nearly zero. In summary: "almost all events are almost equally surprising".

The Shannon-McMillan-Breiman theorem states that AEP also holds for stationary ergodic processes with finite alphabet. Barron extended the Shannon-McMillan-Breiman theorem to processes with infinite alphabets [17].

Theorem B.6. (AEP: Shannon-McMillan-Breiman theorem)
If $H(X)$ is the entropy rate of a finite-valued stationary ergodic process $X = X_1, X_2, \dots, X_n$, then

$$-\frac{1}{n}\log p_X(X_1, X_2, \dots, X_n) \to H(X), \quad \text{w.p.1.} \tag{B.26}$$

Theorem B.7. (Generalized Shannon-McMillan-Breiman theorem [17])
If $h(X)$ is the differential entropy rate of a continuous-valued stationary ergodic process $X = X_1, X_2, \ldots, X_n$ with density $p_X(X_1, X_2, \ldots, X_n)$, then

$$-\frac{1}{n} \log p_X(X_1, X_2, \ldots, X_n) \to h(X), \quad \text{w.p.1.} \tag{B.27}$$

The Shannon-McMillan-Breiman theorem suggests a simple numerical method to compute the entropy rate $H(X)$:

a) Sample a "very long" sequence $x \triangleq (x_1, x_2, \ldots, x_n)$ from $p_X(x)$.

b) The sample sequence entropy rate is an estimate for the entropy rate of X:

$$\hat{H} \triangleq -\frac{1}{n} \log p_X(X_1, X_2, \ldots, X_n) \approx H(X). \tag{B.28}$$

In Chapter 6, we provide more details on this numerical method.

B.2 Channel capacity

B.2.1 Memoryless channels

Shannon has proved [178] that the capacity of a **memoryless** channel with finite **discrete** input alphabet $\mathcal{X}$, finite discrete output alphabet $\mathcal{Y}$ is given by the following expression.

Capacity of (unconstrained) discrete memoryless channel:

$$C \triangleq \max_{p_X} I(X;Y),$$

where the maximum is taken over all input probability mass functions p_X on X. Any input distribution p_X that maximizes the mutual information between X and Y is called a capacity-achieving input distribution. Such a distribution is not necessarily unique; in contrast, the corresponding output distribution is unique.

Remark B.3. (Properties of capacity)

a) $C \geq 0$ since $I(X;Y) \geq 0$.

b) $C \leq \log|\mathcal{X}|$ since $C = \max I(X;Y) \leq \max H(X) = \log|\mathcal{X}|$.

c) $C \leq \log|\mathcal{Y}|$ for the same reason.

Since $I(X;Y)$ is a concave function over a close convex set, a local maximum is a global maximum. From properties (2) and (3), it follows that the maximum is finite, hence we are justified to write maximum instead of supremum in the definition of capacity. The maximum can in principle be found by standard non-linear optimization techniques such as gradient descent. Since the objective function $I(X;Y)$ is concave in p_X, gradient-based algorithms are guaranteed to converge to the global maximum. An efficient alternative to compute the capacity C for discrete memoryless channels is the Blahut-Arimoto algorithm (see Section 7.2).

The input of the channel often needs to fulfill certain conditions. The capacity of a discrete memoryless channel with the requirement that the average cost be less than or equal to some specified number E is given by [178]

Capacity of constrained discrete memoryless channel:

$$C \triangleq \max_{p_X \in P_E} I(X;Y).$$

By P_E, we denote the set of probability mass functions over $\mathcal{X}$ satisfying the constraint $\sum_x p_X(x)e(x) \leq E$.

For only a small number of memoryless channels, a closed-form expression for the capacity is available. Shannon [178] has computed the capacity for the AWGN channel with an average-energy constraint. Assuming that the input power equals P and that the power of the Gaussian noise is σ^2, the capacity is given by

$$C = \frac{1}{2}\log_2\big(1 + \frac{P}{\sigma^2}\big). \tag{B.29}$$

The capacity-achieving input distribution is a zero-mean Gaussian pdf with variance equal to P. If logarithms to the base two are used (as we will generally do), the unit of capacity is bits/symbol. Note that if the input is unconstrained, the capacity of the AWGN channel is infinite.

In Chapter 7, we present a numerical algorithm to compute the capacity (or tight lower bounds on the capacity) of peak-power and/or average-power constrained **continuous** memoryless channels.

B.2.2 Channels with memory

Dobrušin [60] generalized Shannon's result for memoryless channels with finite alphabets to the **continuous-alphabet** case **with memory**.[3]

Capacity of constrained continuous channel with memory:

$$C \triangleq \lim_{n\to\infty} \frac{1}{n} \sup_{p_X \in P_C} I(X;Y),$$

where $X \triangleq (X_1, X_2, \ldots, X_n)$, $Y \triangleq (Y_1, Y_2, \ldots, Y_n)$ and P_C is the set of all probability densities p_X over $\mathcal{X}^n$ satisfying the constraints

$$\frac{1}{n}\sum_{k=1}^{n} \mathrm{E}_{p_X}[e(X_k)] \leq E, \tag{B.30}$$

and

$$|X_k| > A \quad \text{w.p.1} \quad \text{for } k = 1, 2, \ldots, n. \tag{B.31}$$

[3]It is assumed that the channel is *information stable*. This roughly means that the input that maximizes mutual information and its corresponding output behave ergodically. We also assume that there is no feedback.

Appendix C

Coding Theory

Most of the following definitions can be found in any elementary text about coding theory, e.g. [112]. We will follow the exposition in [203].

As is standard, we will only use row vectors. Every code will be a block code, i.e., a code of finite length.[1]

Definition C.1. (Codes)

- **(Block code)**
  An (n, M) block code $\mathcal{C}$ of length n and size M over some alphabet A is a subset $\mathcal{C}$ of size M of A^n, the set of all n-tuples over A. The parameter n is called the length of the block code. An element $\mathbf{x}$ of $\mathcal{C}$ is called a codeword.

- **(Membership indicator function)**
  The membership indicator function I_C of a code $\mathcal{C}$ is defined as

$$I_C : A^n \to \{0, 1\} : \mathbf{x} \to [\mathbf{x} \in \mathcal{C}]. \tag{C.1}$$

- **(Linear code)**
  A block code is linear if $A = \mathbb{F}$ is a field and the set of all codewords forms a k–dimensional subspace (code space) of $\mathbb{F}^n$. Usually the

[1]Most of the time, we will use the shorter "code" instead of "block code". Note that in this thesis, when we talk about a code we always mean a channel code, in contrast to a source code or a line code. See also Sec. 2.1.

field $\mathbb{F}$ is a *finite* field $\mathbb{F}_q$, i.e., the set $\{0, 1, \ldots, q-1\}$ with modulo-q arithmetic. An $[n, k]$ linear code is a linear code of length n and dimension k.

- **(Binary code)**
  A binary code is a linear code with $\mathbb{F} = \mathbb{F}_2$.

□

Definition C.2. (Linear Codes)
Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$.

- **(Generator matrix)**
  The code $\mathcal{C}$ can be defined by a generator matrix $\mathbf{G}$ whose rows form a basis of the code space; therefore $\mathbf{G}$ must have size $k \times n$. Each codeword $\mathbf{x} \in \mathbb{F}_q^n$ can be written as $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}$ with a suitable $\mathbf{u} \in \mathbb{F}_q^k$. $\mathbf{u}$ consists of the information symbols whereas $\mathbf{x}$ consists of the channel symbols. If all information symbols appear one-to-one somewhere in the channel symbols, the encoding is called systematic.

- **(Parity-check matrix)**
  Equivalently, $\mathcal{C}$ can be defined by a parity-check matrix $\mathbf{H}$ whose rows span the space orthogonal to the code space. Such a matrix must fulfill $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$, where $\mathbf{G}$ is a any generator matrix of the code. For every codeword $\mathbf{x} \in \mathcal{C}$ it follows that $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{0}$. Note that the rows of $\mathbf{H}$ need not be a basis of the space orthogonal to the code space, they can also form an over-complete basis. Therefore, $\mathbf{H}$ has n columns and at least $n - k$ rows.

- **(Rate)**
  The rate of the code is defined to be $R \triangleq R(\mathcal{C}) \triangleq k/n$. If $\mathbf{G}$ is a generator matrix, $R \triangleq \#\text{rows}(\mathbf{G})/\#\text{col}(\mathbf{G})$ is the ratio of the number of rows of $\mathbf{G}$ over the number of columns of $\mathbf{G}$. If $\mathbf{H}$ is a parity-check matrix, $R = 1 - \text{rank}(\text{H})/n \geq 1 - \#\text{rows}(\mathbf{H})/\#\text{col}(\mathbf{H})$.

□

Example C.1. (Binary linear block code)
Fig. C.1 shows a small example of linear a code of length $n = 3$, dimension $k = 2$, redundancy $n - k = 1$, rate $R = k/n = 2/3$, and alphabets $\mathcal{U} = \mathcal{X} = \mathbb{F}_2$ with generator and parity-check matrices

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$

$\mathbf{u}$	$\mapsto$	$\mathbf{x} = \mathbf{x}(\mathbf{u})$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(1,0)$	$\mapsto$	$(1,1,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,1)$	$\mapsto$	$(1,0,1)$

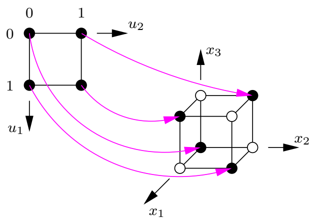


Figure C.1: Example of a simple linear code with $\mathcal{U} = \mathcal{X} = \{0,1\}$, $k = 2$, $n = 3$. Left: Mapping of $\mathbf{u}$ to $\mathbf{x}$ for a binary linear $[3,2,2]$ code. Right: graphical visualization of that mapping.

where the mapping $\mathbf{u} \mapsto \mathbf{x}(\mathbf{u}) = \mathbf{u} \cdot \mathbf{G}$ is given in the table in Fig. C.1.
□

Definition C.3. (LDPC codes)
Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$.

- **(LDPC code)**
The code $\mathcal{C}$ is called a low-density parity-check (LDPC) code if it has a parity-check matrix which has very few ones per row and per column. [2] More precisely, when considering LDPC code families for $n \to \infty$, one usually requires that the number of ones per column and the number of ones per row grow slower than the block length or that these numbers are even bounded.

- **(Regular LDPC code)**
The code $\mathcal{C}$ (defined by a low-density parity-check matrix $\mathbf{H}$) is called a $(w_{\text{col}}, w_{\text{row}})$–regular LDPC code if the Hamming weight of each column of $\mathbf{H}$ equals w_{col} and if the Hamming weight of each row of $\mathbf{H}$ equals w_{row}. The equality $nw_{\text{col}} = mw_{\text{row}}$ relates w_{col} and w_{row}, where m is the number of rows of $\mathbf{H}$.

- **(Irregular LDPC code)**
The code $\mathcal{C}$ (defined by a low-density parity-check matrix $\mathbf{H}$) is called an irregular LDPC code when the Hamming weights of the columns vary and/or the Hamming weights of the rows vary. For more information on (irregular) LDPC codes, see [124].

□

[2]Detailed information about LDPC codes can be found in [1] and [2].

Appendix D

Kernel Methods

In this appendix, we review the most important concepts behind kernel methods; we will closely follow [120]. We refer to [177] [180] for further information.

We also elaborate on how kernels can be computed from graphical models (Section D.5).

D.1 Introduction

Since the mid-1990's, so called "kernel methods" have been drawing much attention. Such methods combine and extend most advantages both of neural networks and of classical linear techniques.

Kernel methods can be used for regression, classification, clustering, and more. A kernel based regression function $\mathbb{R}^n \to \mathbb{R} : y = (y_1, \dots, y_n) \mapsto \hat{x}$ looks as follows:

$$\hat{x} = \sum_{j=1}^{m} w_j \kappa(y, y^{(j)}) + w_0. \tag{D.1}$$

The vectors $y^{(1)}, \dots, y^{(m)} \in \mathbb{R}^n$ are (the y-component of) the training samples and $w_0, \dots, w_m$ are real coefficients (weights). The function

$\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, which must satisfy certain conditions (to be discussed later), is referred to as the *kernel.*

An example of a kernel function is

$$\kappa(y, y') = \exp\left(-\frac{\|y - y'\|}{2\sigma^2}\right). \tag{D.2}$$

Note that:

- The kernel function has no parameters that need to be adjusted.
- The sum in (D.1) runs over all training samples.

The weights w_j can be determined by minimizing the average squared error

$$\text{ASE} = \frac{1}{m}\sum_{\ell=1}^{m}\left(x^{(\ell)} - \sum_{j=1}^{m} w_j \kappa(y^{(\ell)}, y^{(j)}) - w_0\right)^2 \tag{D.3}$$

on the training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, which amounts to a least squares problem. Other cost functions than the ASE are often used in order to force most weights w_j to zero. Since such cost functions are chosen to be convex (in the weights), the optimal weights can be found by efficient algorithms. In consequence (and in sharp contrast to neural networks), kernel methods do not normally need heuristic optimization methods.

The choice of a suitable kernel function—which depends on the application—is the most important step in kernel methods. The domain of the kernel function need not be $\mathbb{R}^n$; in fact, it can be almost any set, including non-numerical data such as strings and images.

The subsequent processing (regression, clustering, classification, ...) is done by optimal (linear or convex) methods and is essentially independent of the application.

D.2 Feature Space Interpretation

A first step of many data analysis methods is to transform the data by some (fixed) mapping

$$\phi : \text{data space} \to \text{feature space} \tag{D.4}$$

before applying some adaptive algorithm (e.g., a neural network). In this chapter, we will always assume that the feature space is a Hilbert space. A *Mercer kernel* (hereafter simply *kernel*) is defined to be the inner product in some feature space:

$$\kappa(z, y) = \langle \phi(z), \phi(y) \rangle \, . \tag{D.5}$$

A main insight behind kernel methods is as follows:

a) With suitable choices of the feature space and of the mapping ϕ, the task (regression, classification, ...) may be solvable by *linear* methods or by *convex* optimization...

b) ...with algorithms that require only the computation of *inner products* in the feature space.

c) Moreover, the inner product in high-dimensional (even infinite-dimensional) feature spaces can often be expressed by quite simple kernel functions.

d) It follows that we do not need to actually evaluate the mapping ϕ; it suffices to evaluate the kernel function κ.

We will work out points 2 and 4 for regression. However, it should be remembered that both the feature space and the mapping ϕ are usually not made explicit; the actual computations in kernel methods involve only the kernel function.

D.2.1 Regression

We consider *nonlinear* regression by means of *linear* regression in feature space. Let S be the original data space and let the feature space be $\mathbb{R}^n$. We wish to find a mapping

$$S \to \mathbb{R} : y \mapsto \phi(y)^T h \tag{D.6}$$

with $h = (h_1, \ldots, h_n)^T \in \mathbb{R}^n$ such that the average squared error (ASE)

$$\text{ASE} = \frac{1}{m} \sum_{\ell=1}^{m} \left| x^{(\ell)} - \phi\Big(y^{(\ell)}\Big)^T h \right|^2 \tag{D.7}$$

on the training data $(x^{(\ell)}, y^{(\ell)})$, $\ell = 1, \ldots, m$, is as small as possible.

It turns out that this problem is essentially the same as the linear regression problem. If we define the column vector $x \triangleq (x^{(1)}, \ldots, x^{(m)})^T$ and the matrix

$$A_\phi \triangleq \Big(\phi(y^{(1)}), \ldots, \phi(y^{(m)})\Big)^T \tag{D.8}$$

$$= \begin{pmatrix} \phi(y^{(1)})_1 & \phi(y^{(1)})_2 & \ldots & \phi(y^{(1)})_n \\ \phi(y^{(2)})_1 & \phi(y^{(2)})_2 & \ldots & \phi(y^{(2)})_n \\ \ldots & & & \ldots \\ \phi(y^{(m)})_1 & \phi(y^{(m)})_2 & \ldots & \phi(y^{(m)})_n \end{pmatrix}, \tag{D.9}$$

the average squared error may be written as

$$m \cdot \text{ASE} = \|x - A_\phi h\|^2 = (x - A_\phi h)^T (x - A_\phi h). \tag{D.10}$$

It then follows that h may be obtained from

$$A_\phi^T A_\phi h = A_\phi^T x. \tag{D.11}$$

However, the point of kernel methods is to avoid solving (D.11)—in fact, to avoid ϕ at all. Instead, we wish to express the regression function (D.6) in terms of the kernel $\kappa(y, z) = \phi(y)^T \phi(z)$.

Theorem D.1. Let g be any function $g : S \to \mathbb{R} : y \mapsto g(y) \triangleq \phi(y)^T h$ with $\phi : S \to \mathbb{R}^n$ and $h \in \mathbb{R}^n$. Then there exists a function $g' : S \to \mathbb{R} : y \mapsto g'(y) \triangleq \phi(y)^T h'$ with the following properties:

a) g' and g give the same values on the training set: $g'(y) = g(y)$ for $y \in \{y^{(1)}, \ldots, y^{(m)}\}$.

b) g' can be written as

$$g'(y) = \sum_{j=1}^{m} w_j \kappa(y, y^{(j)}) \tag{D.12}$$

with real coefficients $w_1, \ldots, w_m$ that depend on the training set.

It follows that the coefficient vector h that minimizes the ASE (D.7) can be replaced by an equivalent vector h' which allows to write the function (D.6) in the form (D.12).

Proof of Theorem D.1: Let V be the subspace of $\mathbb{R}^n$ spanned by $\phi(y^{(1)}), \dots, \phi(y^{(m)})$ and let $V^\perp$ be the orthogonal complement of V in $\mathbb{R}^n$. Let $h = h_V + h_{V^\perp}$ be the decomposition of h into $h_V \in V$ and $h_{V^\perp} \in V^\perp$. Since $\phi(y^{(\ell)})^T h_{V^\perp} = 0$ for $\ell = 1, \dots, m$, we have

$$\phi(y^{(\ell)})^T h = \phi(y^{(\ell)})^T h_V \quad \text{for } \ell = 1, \dots, m. \tag{D.13}$$

Choosing $h' \triangleq h_V$ thus guarantees Property 1 of Theorem D.1. Moreover, by the definition of V, we can write h_V as

$$h_V = \sum_{j=1}^{m} w_j \phi(y^{(j)}) \tag{D.14}$$

with $w_j \in \mathbb{R}$. Thus

$$\begin{aligned} g'(y) &= \phi(y)^T h_V && \text{(D.15)} \\ &= \phi(y)^T \sum_{j=1}^{m} w_j \phi(y^{(j)}) && \text{(D.16)} \\ &= \sum_{j=1}^{m} w_j \phi(y)^T \phi(y^{(j)}) && \text{(D.17)} \\ &= \sum_{j=1}^{m} w_j \kappa(y, y^{(j)}). && \text{(D.18)} \end{aligned}$$

□

D.3 Support Vector Machines

If the regression function (D.1) is trained to minimize the ASE on the training data, then virtually all weights w_j will probably be nonzero. Replacing the ASE by other cost functions opens the possibility to force a large fraction of the weights to zero. Those training samples $y^{(j)}$ for which w_j is nonzero are called *support vectors*. This approach can also be carried out for classification.

Such *support vector machines* are attractive because the regression (or classification) function can be evaluated more efficiently. The determination of the optimal weights takes a little more effort, but is still a convex problem for which efficient algorithms are available.

D.4 On Kernels

Kernels can be characterized as follows. Recall that a symmetric real $n \times n$ matrix A is *positive semi-definite* if $x^T Ax \geq 0$ for every $x \in \mathbb{R}^n$.

Theorem D.2 (Characterization of Kernels)**.** A function

$$\kappa : S \times S \to \mathbb{R} \tag{D.19}$$

(which is either continuous or has a finite domain S) is a kernel if and only if the following two conditions hold:

a) k is symmetric (i.e., $\kappa(z, y) = \kappa(y, z)$)

b) for every finite subset $\{y_1, \dots, y_n\}$ of S, the matrix

$$\begin{pmatrix} \kappa(y_1, y_1) & \kappa(y_1, y_2) & \dots & \kappa(y_1, y_n) \\ \kappa(y_2, y_1) & \kappa(y_2, y_2) & \dots & \kappa(y_2, y_n) \\ \dots & & & \dots \\ \kappa(y_n, y_1) & \dots & & \kappa(y_n, y_n) \end{pmatrix} \tag{D.20}$$

is positive semi-definite.

(The proof is not hard, but we omit it.)

So far, we have encountered only one explicit kernel function: the Gaussian kernel (D.2). Another kernel is the function

$$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} : (y, z) \mapsto y^T Az \tag{D.21}$$

where A is a symmetric positive semi-definite $n \times n$ matrix over $\mathbb{R}$. (The proof requires some standard linear algebra.)

More kernels can be constructed from the following theorem.

Theorem D.3. Let κ_1 and κ_2 be kernels over $S \times S$. The following functions are also kernels over $S \times S$:

a) $\kappa(y,z) = \kappa_1(y,z) + \kappa_2(y,z)$

b) $\kappa(y,z) = \alpha\kappa_1(y,z)$ for any positive $\alpha \in \mathbb{R}$

c) $\kappa(y,z) = \kappa_1(y,z)\kappa_2(y,z)$

d) $\kappa(y,z) = f(y)f(z)$ for any function $f : S \to \mathbb{R}$

e) $\kappa(y,z) = \kappa_3(g(y), g(z))$ for any kernel κ_3 over $\mathbb{R}^n \times \mathbb{R}^n$ and any function $g : S \to \mathbb{R}^n$.

f) $\kappa(y,z) = p(\kappa_1(y,z))$ for every polynomial $p(x)$ with positive coefficients.

The proof is left as an exercise.

D.5 Kernels from Graphical Models

In this section, we investigate how kernels can be computed from a graphical model.[1]

Suppose that we wish to process—i.e., cluster, compress, classify, etc.—a data set $\mathcal{D} = \{\hat{y}_1, \ldots, \hat{y}_N\}$ by means of some kernel method.

Assume that we have determined a graphical model $p(y, x|\theta)$ that to some extend is able to "explain" the data $\mathcal{D}$; by θ we denote the parameter vector of the model, x is the vector of hidden variables, and y are the observed variables. For example, the data $\mathcal{D}$ may be a set of EEG signals, which we model by a multi-channel ARMA model $p(y, x|\theta)$.

We wish to incorporate the available model $p(y, x|\theta)$ in our kernel method. As is clear from the exposition in the previous sections, the model $p(y, x|\theta)$ should then somehow be encoded in the *kernel*, since the kernel is the sole element in a kernel method that depends on the application at hand. Once the kernel is specified, the subsequent processing (clustering, compression, classification, etc.) is accomplished by standard optimization tools (independently of the application) [180] [177].

In the following, we will explore several ways to derive kernels from graphical models. We will outline how the message-passing techniques of Chapter 4 may be used in this context.

[1]The extension to a *family* of graphical models is straightforward.

D.5.1 Probabilistic Kernel

A first option is the so-called *probabilistic kernel* [180]:

$$\kappa(\hat{y}_i, \hat{y}_j) \triangleq \sum_x p(\hat{y}_i, x|\hat{\theta}) p(\hat{y}_j, x|\hat{\theta}), \tag{D.22}$$

where the parameters $\hat{\theta}$ are obtained from the whole data set $\mathcal{D}$, e.g., by ML-estimation:

$$\begin{aligned} \hat{\theta}^{\text{ML,tot}} &\triangleq \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(\hat{y}_i|\theta) &\text{(D.23)}\\ &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} \sum_x p(\hat{y}_i, x|\theta). &\text{(D.24)}\end{aligned}$$

It is easy to verify that (D.22) is indeed a kernel (cf. Theorem D.3).

D.5.2 Product Kernel

An alternative is the so-called *product-kernel* [91]. Each data point $\hat{y}_i$ is mapped unto a probabilistic model $p(y|\hat{y}_i)$ defined as:

$$p(y|\hat{y}_i) \triangleq \sum_x p(y|x, \hat{\theta}) p(x|\hat{\theta}, \hat{y}_i), \tag{D.25}$$

where the parameters $\hat{\theta}$ is estimated by means of the sample $\hat{y}_i$, e.g., by ML estimation:

$$\begin{aligned} \hat{\theta}^{\text{ML}} &\triangleq \underset{\theta}{\operatorname{argmax}}\, p(\hat{y}_i|\theta) &\text{(D.26)}\\ &= \underset{\theta}{\operatorname{argmax}} \sum_x p(\hat{y}_i, x|\theta). &\text{(D.27)}\end{aligned}$$

The product-kernel is computed as follows:

$$\kappa(\hat{y}_i, \hat{y}_j) \triangleq \sum_y p(y|\hat{y}_i) p(y|\hat{y}_j). \tag{D.28}$$

D.5.3 Fisher Kernel

A third approach is the so-called *Fisher kernel*, defined as [89]:

$$\kappa(\hat{y}_i, \hat{y}_j) \overset{\triangle}{=} \nabla_\theta^T \log p(\hat{y}_i|\hat{\theta}) \mathbf{F}^{-1}(\hat{\theta}) \nabla_\theta \log p(\hat{y}_j|\hat{\theta}), \tag{D.29}$$

where $p(y|\theta)$ is defined as:

$$p(y|\theta) \overset{\triangle}{=} \sum_x p(y, x|\theta), \tag{D.30}$$

and the estimate $\hat{\theta}$ is obtained from the whole data set $\mathcal{D}$, e.g., by ML-estimation (D.24). The matrix $\mathbf{F}(\theta)$ is the Fisher information matrix of $p(y|\theta)$ (cf. Section 5.2.1).

D.5.4 Discussion

Some remarks:[2]

- The three types of kernels we listed in the above involve sums and/or integrals and maximizations. In particular:

  a) The expressions (D.22), (D.25), and (D.30) involve summation/integration over x.

  b) The expression (D.28) involves summation/integration over y.

  c) The expressions (D.24) and (D.27) involve maximization over θ.

  In principle, the summations/integrations and the maximizations may be carried out (exactly) by applying the sum-product algorithm and max-product algorithm respectively on a suitable (cycle-free) factor graph.

  For example, if the system at hand is a linear system perturbed by Gaussian noise sources, the sum-product algorithm boils down to Kalman recursions (cf. Appendix H).

  If the variables X are discrete, the sum-product algorithm may reduce to applying the BCJR-algorithm [15] on a trellis of the system.

[2]Some of our observations may be novel.

- The sum/max-product algorithm may, however, lead to intractable expressions, more precisely:

  a) sums with an unwieldy number of terms,

  b) intractable integrals,

  c) intractable maximizations.

  Intractable sums (Problem 1) may be computed approximately by applying the (iterative) sum-product algorithm on a cyclic graph.

  Intractable integrals (Problem 2) may be approximated by applying the (iterative) sum-product algorithm on a cyclic graph in combination with numerical integration or Monte-Carlo methods (cf. Chapter 4).

  Intractable maximizations (Problem 3) may be performed approximately, e.g., by ICM, EM, or gradient methods (cf. Chapter 4).

  Note, however, that such approximations may not always lead to a kernel! Therefore, certain approximations are *not* allowed (if we wish to derive kernels). We investigate each of the three kernels in more detail:

  – **Probabilistic kernel:**
    If one computes the sum/integral in (D.22) approximately by the iterative sum-product algorithm, one does *not* obtain a kernel in general. However, the sum/integral may be evaluated by means of Monte-Carlo integration.

    The maximization (D.24) may be carried out by any estimation method as for example EM, ICM, and gradient methods (cf. Chapter 4).

  – **Product kernel:**
    In order to obtain a kernel, $p(y|\hat{y}_i)$ (cf. (D.28)) may be *any* function, i.e., it does not need to be defined as in (D.25). Therefore, the sum/integral in (D.27) may be evaluated by any approximative method; moreover, the parameter $\hat{\theta}$ may be determined by any method.

    If the sum/integral in (D.28) is computed by iterative sum-product message-passing, one does not obtain a kernel; however, (D.28) may be evaluated by numerical or Monte-Carlo integration.

- **Fisher kernel:**
  In order to obtain a kernel, $p(y|\theta)$ may be any function and $\mathbf{F}^{-1}(\hat{\theta})$ may be any positive definite matrix. Therefore, one has the freedom to use approximative methods to carry out the sum/integral in (D.30), resulting in an approximation $q(y|\theta)$ of $p(y|\theta)$ (D.30). One may then obtain an approximation of $\mathbf{F}(\hat{\theta})$ as

$$\tilde{\mathbf{F}}_p(\hat{\theta}) \triangleq \mathrm{E}_{p(y|\theta)}\left[\nabla_\theta q(y|\theta)\nabla_\theta^T q(y|\theta)\right] \tag{D.31}$$

  or as

$$\tilde{\mathbf{F}}_q(\hat{\theta}) \triangleq \mathrm{E}_{q(y|\theta)}\left[\nabla_\theta q(y|\theta)\nabla_\theta^T q(y|\theta)\right]. \tag{D.32}$$

  The matrices $\tilde{\mathbf{F}}_p$ and $\tilde{\mathbf{F}}_q$ are guaranteed to be positive semi-definite—also in the case where the expectations in (D.31) and (D.32) are computed by numerical integration or Monte-Carlo methods. If those two matrices are moreover positive definite, they can be used to construct kernels:

$$\kappa_p(\hat{y}_i, \hat{y}_j) \triangleq \nabla_\theta^T \log q(\hat{y}_i|\hat{\theta})\tilde{\mathbf{F}}_p^{-1}(\hat{\theta})\nabla_\theta \log q(\hat{y}_j|\hat{\theta}), \tag{D.33}$$

  and

$$\kappa_q(\hat{y}_i, \hat{y}_j) \triangleq \nabla_\theta^T \log q(\hat{y}_i|\hat{\theta})\tilde{\mathbf{F}}_q^{-1}(\hat{\theta})\nabla_\theta \log q(\hat{y}_j|\hat{\theta}). \tag{D.34}$$

  It is noteworthy that the matrices $\tilde{\mathbf{F}}_p$ and $\tilde{\mathbf{F}}_q$ are *not* guaranteed to be positive semi-definite if the expectation in (D.31) and (D.32) is carried out by the iterative sum-product algorithm.
  Note also that in order to obtain a kernel, one can use any method to estimate $\hat{\theta}$.

- It is straightforward to combine the three different approaches. In addition, it is not very difficult to come up with variations on the same theme.

We have outlined three different strategies to generate kernels from graphical models. We pointed out how message-passing techniques can be used in this context. However, several important question remains unanswered: which of the three classes of kernels should we use after all? Is any of the above kernels optimal in some sense? A definite answer to such questions seems not to have been formulated yet in the literature, and goes beyond the scope of this exposition.

Appendix E

Neural Networks

This section reviews the main ideas behind feed-forward neural networks; we will closely follow [120].

We will also explain how the back-propagation algorithm, which is widely used to train feed-forward neural networks, can be regarded as message passing on factor graphs.

E.1 Multi-Layer Perceptron

A *perceptron* is a mapping

$$\mathbb{R}^n \to \mathbb{R} : (y_1, \ldots, y_n) \mapsto g_{\mathrm{s}}\left(w_0 + \sum_{k=1}^{n} w_k y_k\right) \tag{E.1}$$

with real parameters $w_0, \ldots, w_1$ and where g_{s} is the step function

$$g_{\mathrm{s}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0. \end{cases} \tag{E.2}$$

Such perceptrons are classifiers with two decision regions that are separated by a plane.

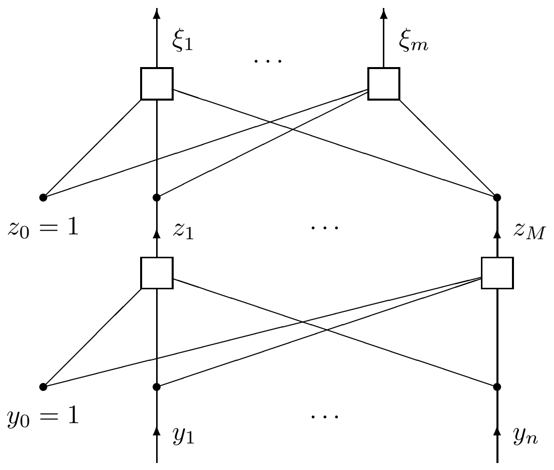


Figure E.1: Two-layer perceptron

We will proceed directly to multi-layer perceptrons. A *two-layer perceptron* is a mapping

$$\mathbb{R}^n \to \mathbb{R}^m : (y_1, \ldots, y_n) \mapsto (\xi_1, \ldots, \xi_m) \tag{E.3}$$

with

$$\xi_k = g_{\text{out}}\left(w_{k0}^{(2)} + \sum_{j=1}^{M} w_{kj}^{(2)}\, g\left(w_{j0}^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} y_i \right)\right) \tag{E.4}$$

with real parameters $w_{ji}^{(1)}$ and $w_{kj}^{(2)}$ and with functions g and g_{out} as discussed below. With $y_0 \triangleq 1$ and with

$$z_j \triangleq g\left(\sum_{i=0}^{n} w_{ji}^{(1)} y_i \right) \tag{E.5}$$

for $j = 1, \ldots, M$ and with $z_0 \triangleq 1$, the mapping (E.4) becomes

$$\xi_k = g_{\text{out}}\left(\sum_{j=0}^{M} w_{kj}^{(2)} z_j \right). \tag{E.6}$$

The structure of such a two-layer perceptron is illustrated in Fig. E.1.

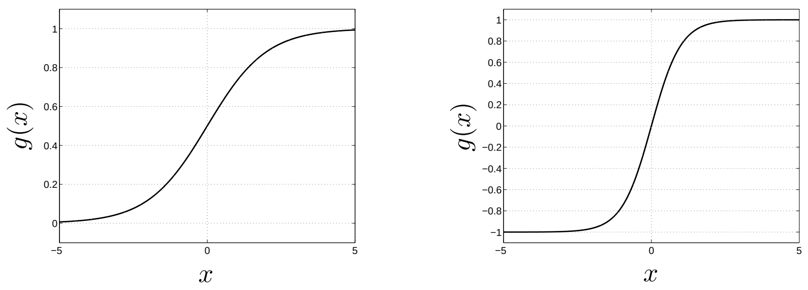


Figure E.2: Sigmoid (left) and tanh (right).

The function g in (E.4) and (E.5) is usually either the *logistic sigmoid* function

$$g(x) = \frac{1}{1 + e^{-x}} \tag{E.7}$$

(which may be viewed as a “soft” version of the step function (E.2)) or the hyperbolic tangent function

$$g(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{E.8}$$

see Fig. E.2. The function g_{out} in (E.4) is usually one of the following functions: the sigmoid (E.7) or the hyperbolic tangent (E.8) or simply

$$g(x) = x. \tag{E.9}$$

Note that g_{out} and g need not be the same function. In contrast to the step function (E.2), the functions (E.7)–(E.9) are everywhere differentiable.

The variables $z_1, \dots, z_M$ are called *hidden variables* (or *hidden nodes*). In (E.4) and in Fig. E.1, we have two layers of weights and one layer of hidden nodes. The generalization to networks with more layers is straightforward.

Multi-layer perceptrons (with at least two layers of weights) can represent essentially any continuous mapping to arbitrary accuracy. The training of such networks (i.e., the determination of the weights from input/output samples) is a nonlinear optimization problem that can be solved with a variety of methods, but none of these methods is guaranteed to find optimal values for the weights.

For many applications, two layers of weights suffice. Additional layers are sometimes used to build special properties (e.g., symmetries) into the network.

Two-layer perceptrons can be used both for regression and for classification. In the former case (regression), the network is (e.g.) supposed to represent the function

$$\mathbb{R}^n \to \mathbb{R}^m : (y_1, \ldots, y_n) \mapsto \mathrm{E}[X_1, \ldots, X_m \mid Y_1 = y_1, \ldots, Y_n = y_n] \tag{E.10}$$

for some real random variables $X_1, \ldots, X_m$. In this case, the output function g_{out} is usually chosen to be the linear function (E.9). With this choice of g_{out}, it is obvious from (E.6) that the determination of the second-layer weights $w_{kj}^{(2)}$ (for fixed first-layer weights $w_{ji}^{(1)}$) is a least-squares problem; in this case, only the first-layer weights $w_{ji}^{(1)}$ need to be determined by nonlinear optimization methods.

For classification applications, the network is (e.g.) supposed to represent the function

$$\begin{aligned}&\mathbb{R}^n \to \mathbb{R}^m : \; (y_1, \ldots, y_n) \; \mapsto \\ &\Big(P(\tilde{X} \in \mathcal{C}_1 \,|\, Y_1 = y_1, \ldots, Y_n = y_n), \ldots, P(\tilde{X} \in \mathcal{C}_m \,|\, Y_1 = y_1, \ldots, Y_n = y_n)\Big)\end{aligned} \tag{E.11}$$

where $\tilde{X}$ is some random variable and where $\mathcal{C}_1, \ldots, \mathcal{C}_m$ are the different classes. In this case, a natural choice for g_{out} is the sigmoid function (E.7).

Training means the determination of the weights from training data

$$\begin{array}{l}(x_1^{(1)}, \ldots, x_m^{(1)}), (y_1^{(1)}, \ldots, y_n^{(1)}) \\ \ldots \\ (x_1^{(N)}, \ldots, x_m^{(N)}), (y_1^{(N)}, \ldots, y_n^{(N)})\end{array} \tag{E.12}$$

where N is the number of samples. For regression as in (E.10), the natural error function (to be minimized on the training data) is the average squared error

$$\mathrm{ASE} = \frac{1}{N} \sum_{\ell=1}^{N} \sum_{k=1}^{m} \left| x_k^{(\ell)} - \xi_k(y^{(\ell)}) \right|^2 . \tag{E.13}$$

For classification (conditional probability estimation) as in (E.11), we define the random variables X_k, $k = 1, \ldots, m$, as the indicator variables for the classes $\mathcal{C}_k$:

$$X_k = \begin{cases} 1 & \text{if } \tilde{X} \in \mathcal{C}_k \\ 0 & \text{else.} \end{cases} \tag{E.14}$$

Note that, in this case, the training data (E.12) is assumed to contain samples of X_k, $k = 1, \ldots, m$ (not $\tilde{X}$). The error function (E.13) can still be used: it is an easy exercise to prove that, in the limit of infinitely many training data, the functions $\xi_k(y_1, \ldots, y_n)$ that minimize (E.13) will (almost surely) converge to the desired function (E.11). (Nevertheless, other error functions may work better in this case.)

E.2 Back-Propagation of Derivatives

Let E denote some error function (such as, e.g., (E.13)), which (for fixed training data) is a nonlinear function of the weights. A key feature of multi-layer perceptrons is that the derivative $\frac{\partial E}{\partial w}$ can be efficiently computed for all weights w in the network. With these derivatives, we can optimize the weights by a steepest-descent method; moreover, some more advanced optimization methods (such as the conjugate gradient algorithm) need these derivatives as well.

We describe the computation of these derivatives for the two-layer network of Fig. E.1; the extension to networks with more layers is obvious. We begin with the top layer. Let

$$\alpha_k \triangleq \sum_{j=0}^{M} w_{kj}^{(2)} z_j \tag{E.15}$$

so that (E.6) becomes

$$\xi_k = g_{\text{out}}(\alpha_k). \tag{E.16}$$

We then have

$$\begin{aligned} \frac{\partial E}{\partial w_{kj}^{(2)}} &= \frac{\partial E}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial w_{kj}^{(2)}} && \text{(E.17)} \\ &= \frac{\partial E}{\partial \alpha_k} z_j && \text{(E.18)} \end{aligned}$$

with

$$\frac{\partial E}{\partial \alpha_k} = g'_{\text{out}}(\alpha_k)\,\frac{\partial E}{\partial \xi_k}. \tag{E.19}$$

Now to the bottom layer. Let

$$\beta_j \triangleq \sum_{i=0}^{n} w_{ji}^{(1)} y_i \tag{E.20}$$

so that (E.5) becomes

$$z_j = g(\beta_j). \tag{E.21}$$

We then have

$$\begin{aligned}
\frac{\partial E}{\partial w_{ji}^{(1)}} &= \frac{\partial E}{\partial \beta_j}\,\frac{\partial \beta_j}{\partial w_{ji}^{(1)}} && \text{(E.22)}\\
&= \frac{\partial E}{\partial \beta_j}\,y_i, && \text{(E.23)}
\end{aligned}$$

and the key to all is

$$\begin{aligned}
\frac{\partial E}{\partial \beta_j} &= \sum_{k=1}^{m} \frac{\partial E}{\partial \alpha_k}\,\frac{\partial \alpha_k}{\partial \beta_j} && \text{(E.24)}\\
&= \frac{\partial z_j}{\partial \beta_j} \sum_{k=1}^{m} \frac{\partial E}{\partial \alpha_k}\,\frac{\partial \alpha_k}{\partial z_j} && \text{(E.25)}\\
&= g'(\beta_j) \sum_{k=1}^{m} w_{kj}^{(2)}\,\frac{\partial E}{\partial \alpha_k} && \text{(E.26)}
\end{aligned}$$

All these quantities can now be computed as follows. For fixed inputs and fixed weights, we first compute all β_j and z_j and then all α_k and ξ_k by a forward (bottom-to-top) pass through the network. We then compute, first, all derivatives $\frac{\partial E}{\partial \alpha_k}$ by (E.19), and second, all derivatives $\frac{\partial E}{\partial \beta_j}$ by (E.26) by a backward (top-to-bottom) pass through the network. The desired derivatives $\frac{\partial E}{\partial w_{kj}^{(2)}}$ and $\frac{\partial E}{\partial w_{ji}^{(1)}}$ can then be obtained from (E.18) and (E.23), respectively.

E.3 Back-Propagation of Derivatives as Message Passing

In Section 4.8.1, we claim that, if one applies the generic rules for computing derivatives of sum-product messages on a factor graph that represents a feed-forward neural network, one obtains the back-propagation algorithm of Section E.2. Here, we work out the details.

Fig. E.3 depicts a factor graph that represents the feed-forward neural network of Fig. E.1; more precisely, the graph represents the function:

$$f(\xi, y, w) \stackrel{\triangle}{=} \delta(\xi - \xi(y, w)). \tag{E.27}$$

The graph does not incorporate the error function $E(x, \xi)$ (e.g., the ASE (E.13)) and the training data $\big\{(x^{(\ell)}, y^{(\ell)})\big\}_{\ell=1}^{N}$.

We handle the error function $E(x, \xi)$ and the training data as follows. We interpret the training of the weights as an *estimation problem.* In this setting, the random variables ξ are observed through the (artificial) *noisy channel*:

$$\tilde{p}(x|\xi) \quad \stackrel{\triangle}{\propto} \quad e^{-E(x,\xi)}. \tag{E.28}$$

We write $\tilde{p}(\cdot|\cdot)$ instead of $p(\cdot|\cdot)$ since the observation model is artificial. Since the error function $E(x, \xi)$ most often has the form

$$E(x, \xi) \quad = \quad \sum_{k=1}^{m} E_k(x_k, \xi_k), \tag{E.29}$$

we can rewrite (E.28) as:

$$\begin{aligned} \tilde{p}(x|\xi) \quad &\stackrel{\triangle}{\propto} \quad e^{-E(x,\xi)} && \text{(E.30)} \\ &= \quad \prod_{k=1}^{m} e^{-E_k(x_k,\xi_k)} && \text{(E.31)} \\ &\stackrel{\triangle}{\propto} \quad \prod_{k=1}^{m} \tilde{p}(x_k|\xi_k). && \text{(E.32)} \end{aligned}$$

The random variables Y are *directly* observed.

The corresponding factor graph is shown in Fig. E.4. The graph represents the function:

$$\begin{aligned} p(x, \xi | y, w) &\stackrel{\triangle}{=} \prod_{\ell=1}^{N} f(\xi^{(\ell)}, y^{(\ell)}, w) \prod_{k=1}^{m} \tilde{p}(x_k^{(\ell)} | \xi_k^{(\ell)}) && \text{(E.33)} \\ &= \prod_{\ell=1}^{N} \delta(\xi^{(\ell)} - \xi(y^{(\ell)}, w)) \prod_{k=1}^{m} \tilde{p}(x_k^{(\ell)} | \xi_k^{(\ell)}). && \text{(E.34)} \end{aligned}$$

The boxes denoted by “FF NN” are detailed in Fig. E.3; they stand for the function $f(\xi, y, w)$. From the observations of ξ and Y (i.e., the training data $\{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^{N}$), we wish to estimate the random variables W (i.e., the weights in the feed-forward neural network) as:

$$\begin{aligned} \hat{w} &\stackrel{\triangle}{=} \operatorname*{argmin}_{w} E(x, \xi(y, w)) && \text{(E.35)} \\ &= \operatorname*{argmax}_{w} \int_{\xi} \log \tilde{p}(x|\xi) f(\xi, y, w) d\xi && \text{(E.36)} \\ &\stackrel{\triangle}{=} \operatorname*{argmax}_{w} \mathrm{E}_{f(\xi, y, w)} \left[\log \tilde{p}(x|\xi) \right], && \text{(E.37)} \end{aligned}$$

where x and y in (E.37) are given by the training data. The expectation in (E.37) can be carried out by means of the sum-product algorithm, which in this case is a trivial computation. The maximization in (E.37) can in principle be accomplished by applying the max-product algorithm on the graph Fig. E.4, where the boxes “FF NN” are considered as compound nodes, i.e., they are not replaced by the graph of Fig. E.3. From this perspective, the graph of Fig. E.4 is cycle-free, and applying the max-product algorithm on Fig. E.4 leads to the mode (E.37).

What we have done so far is to rewrite the error function E as an expectation and to show that the training of a feed-forward neural network can formally be carried out by sum/max-product message-passing on a suitable factor graph. So far, the message-passing view does probably not buy us much. The key point is, however, that the computation (E.37) is usually *intractable*. Therefore, we need to resort to approximate techniques. And at this point, the message-passing view is arguably useful: on the graph of Fig. E.4 (or any transformed factor graph), one may apply the **message-passing methods** of Chapter 4 such as particle methods (e.g., MCMC), EM, ICM, gradient methods (e.g., steepest descent or natural-gradient descent), or combinations of those methods.

The factor graph of Fig. E.4 is also interesting for a different reason. The graphs suggest us to draw **additional nodes** (as illustrated in Fig. E.5), in particular:

- a node that represents a **prior** $p_W(w)$ for the weights w.
- nodes that represent a **noisy channel** $p(y|z)$ for the observation of Y. (In Fig. E.5, we have depicted a memoryless channel; of course, more sophisticated observation models could be handled.)
- nodes that represent a **prior** $p_Z(z)$ for the random variables Z.
- nodes that represent a **noisy channel** $p(\chi|\xi)$ for the observation of ξ.

The graph of Fig. E.5 represents the function

$$\begin{aligned}p(x,\xi,z,y,w) \stackrel{\triangle}{=} p_Z(z)p_W(w)\prod_{i=1}^{N}\Bigg[&\delta(\xi^{(i)}-\xi(z^{(i)},w))\\ &\cdot\Big(\prod_{j=1}^{m}\tilde{p}(x_j^{(i)}|\chi_j^{(i)})p(\chi_j^{(i)}|\xi_j^{(i)})\Big)\Big(\prod_{j=1}^{n}p(y_j^{(i)}|z_j^{(i)})\Big)\Bigg].\end{aligned} \tag{E.38}$$

The training data is most often noisy, which can be modeled *explicitly* by including the observation model $p(y|z)$ and $p(\chi|\xi)$ in the factor graph, as in Fig. E.5. In the neural-networks literature, the training data is most often pre-processed in an ad-hoc manner. If we model the observation process explicitly, the pre-processing follows directly by message passing on the graph of Fig. E.5, as shown in Fig. E.6. Also here, one may use the catalogue of message-passing tools presented in Chapter 4, as one wishes.

The computation (E.37) now takes the form:

$$\begin{aligned}\hat{w} &\stackrel{\triangle}{=} \operatorname*{argmin}_w \int_{z,\chi}\left[E(\chi,\xi(z,w))p(x|\chi)p(y|z)p_W(w)p_Z(z)dzd\chi\right] \quad \text{(E.39)}\\ &= \operatorname*{argmax}_w \Bigg[\Bigg[\int_{\xi,z,\chi}\log\tilde{p}(\chi|\xi)f(\xi,z,w)p(x|\chi)\end{aligned}$$

$$\left. \cdot p(y|z)p_W(w)p_Z(z)dzd\chi d\xi \right] \tag{E.40}$$

$$\overset{\triangle}{=} \operatorname*{argmax}_{w} \mathrm{E}_{g(x,\xi,\chi,y,z,w)} \left[\log \tilde{p}(\chi|\xi)\right], \tag{E.41}$$

where

$$g(x,\xi,\chi,y,z,w) \overset{\triangle}{=} f(\xi,z,w)p(x|\chi)p(y|z)p_W(w)p_Z(z). \tag{E.42}$$

In the following, we will not explore the graphs of Fig. E.5 and Fig. E.6 any further. Instead, we will focus our attention on the back-propagation algorithm outlined in E.2: we will show that, if one applies the generic rules for computing gradients of sum-product messages (on the factor graphs of Fig. E.3 and Fig. E.4), one obtains the back-propagation algorithm of Section E.2.

If we wish to determine the weights W (cf. (E.35)–(E.37)) by a gradient method, the derivatives of the sum-product messages along the W-edges are required. In Section 4.8.1, we explain how derivatives of generic sum-product messages can be computed. We will use the update rules of Section 4.8.1 in the following.

As an illustration, suppose we wish to compute the derivative of the sum-product message along the edge $W_{11}^{(2)}$ coming from the incident multiply node (cf. Fig. E.3). We will derive the required computations from Fig. E.7, which shows the subgraph of Fig. E.3 that is relevant for the computations at hand.

The edge w is incident to a multiply node. A multiply node is a deterministic node, hence, we apply rule (4.115), resulting in:

$$\nabla_{w_1}\mu_{\boxtimes\to w_1}(w_1) = \nabla_{w_1}h(\hat{z}_1,w_1)\,\nabla_{\gamma_1}\,\mu_{\gamma_1\to\boxtimes}(\gamma_1)\big|_{\gamma_1=h(\hat{z}_1,w_1)}, \tag{E.43}$$

$$= \hat{z}_1\,\nabla_{\gamma_1}\,\mu_{\gamma_1\to\boxtimes}(\gamma_1)\big|_{\gamma_1=\hat{z}_1\cdot w_1}, \tag{E.44}$$

where we used the short-hand notation W_1 for $W_{11}^{(2)}$, $h(z_1,w_1) \overset{\triangle}{=} z_1 \cdot w_1$, and the incoming message $\mu_{z_1\to\boxtimes}(z_1)$ is represented by the estimate $\hat{z}_1$. Note that the derivative $\nabla_{\gamma_1}\,\mu_{\gamma_1\to\boxtimes}(\gamma_1)$ is required. This derivative is computed similarly. We apply the rule (4.115) to the incident addition node, resulting in:

$$\nabla_{\gamma_1}\,\mu_{\gamma_1\to\boxtimes}(\gamma_1) = \nabla_{\alpha_1}\,\mu_{\alpha_1\to\boxplus}(\alpha_1)\big|_{\alpha_1=h(\hat{\gamma}_0+\gamma_1+\hat{\gamma}_2+\cdots+\hat{\gamma}_M)}. \tag{E.45}$$

The derivative $\nabla_{\alpha_1} \mu_{\alpha_1 \to \boxplus}(\alpha_1)$ is computed by applying the rule (4.115) to the node g_{out}, resulting in:

$$\nabla_{\alpha_1} \mu_{\alpha_1 \to \boxplus}(\alpha_1) \quad = \quad \nabla_{\alpha_1} g_{\text{out}}(\alpha_1)\big|_{\alpha_1} \nabla_{\xi_1} \mu_{\xi_1 \to g}(\xi_1)\big|_{\xi_1 = g_{\text{out}}(\alpha_1)}. \quad \text{(E.46)}$$

The message $\nabla_{\xi_1} \mu_{\xi_1 \to g}(\xi_1)$ is given by:

$$\begin{aligned} \nabla_{\xi_1} \mu_{\xi_1 \to g}(\xi_1) &\stackrel{\triangle}{=} \nabla_{\xi_1} \log \tilde{p}(x_1|\xi_1) && \text{(E.47)} \\ &= -\nabla_{\xi_1} E_k(x_1, \xi_1), && \text{(E.48)} \end{aligned}$$

i.e., the derivative of the *logarithm* of $\tilde{p}(x_1|\xi_1)$ (cf. (E.37)), which is also equal to minus the derivative of the error function $E_k(x_1, \xi_1)$. By combining the expressions (E.44)–(E.48), one obtains (E.18)–(E.19). Along similar lines, one can derive (E.23)–(E.26).

In conclusion, we have shown how the back-propagation rules (E.18)–(E.26) follow by mechanically applying the generic rules of Section 4.8.1.

As we pointed out, the required derivatives can be computed in two steps. The first step is a bottom-top sweep, in which, for fixed inputs and fixed weights, all β_j and z_j and then all α_k and ξ_k are updated. This (bottom-top) sweep is depicted in Fig. E.8, where the message-passing procedure inside the boxes "FF NN" is detailed in Fig. E.10.

In a second step, all derivatives $\frac{\partial E}{\partial \alpha_k}$ are computed by (E.19), and all derivatives $\frac{\partial E}{\partial \beta_j}$ are computed by (E.26) by a backward (top-to-bottom) pass through the network. The desired derivatives $\frac{\partial E}{\partial w_{kj}^{(2)}}$ and $\frac{\partial E}{\partial w_{ji}^{(1)}}$ can then be obtained from (E.18) and (E.23), respectively. The message-passing procedure of the second step is depicted in Fig. E.9, where the message passing inside the boxes "FF NN" is detailed in Fig. E.11.

Remark E.1. (Scheduling)
The update of the weights can be scheduled in several ways. One may for example apply stochastic approximation (see Section 4.8.3).

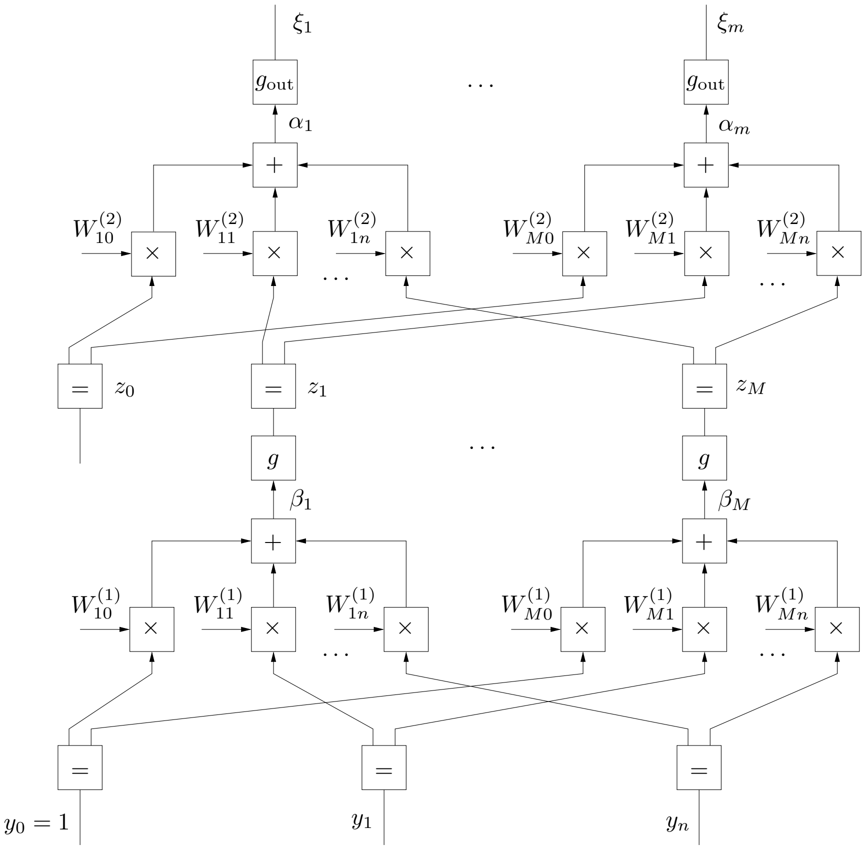


Figure E.3: Factor graph representing a feed-forward neural network.

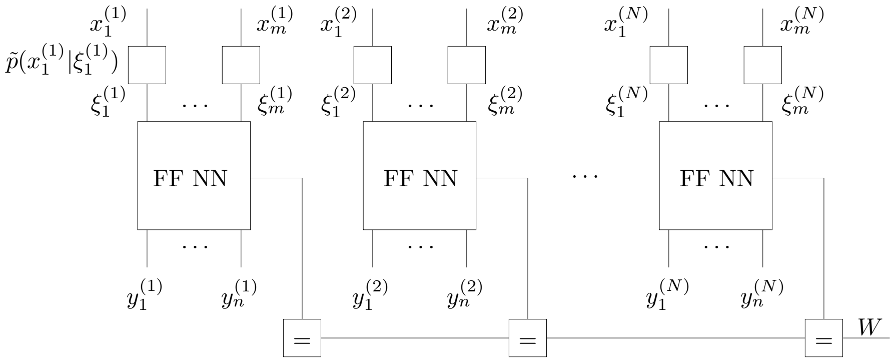


Figure E.4: Training of a feed-forward neural network viewed as an estimation problem.

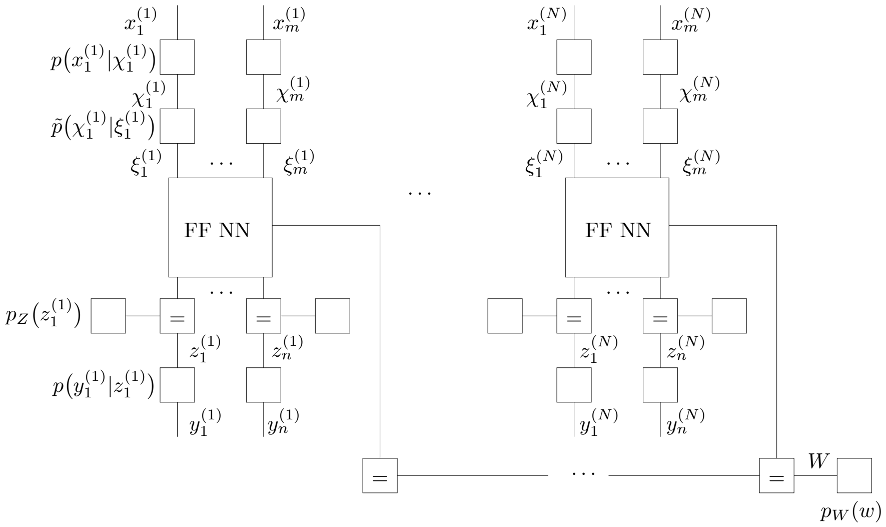


Figure E.5: Additional nodes.

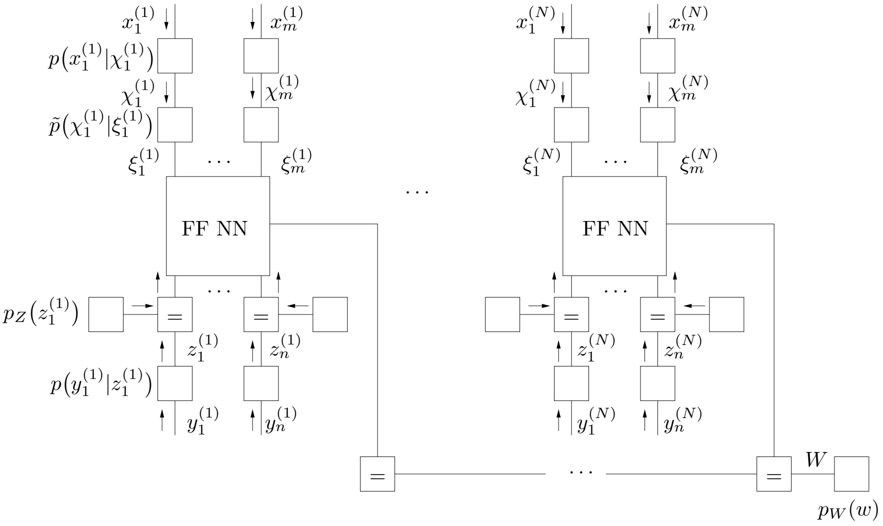


Figure E.6: Pre-processing by message passing.

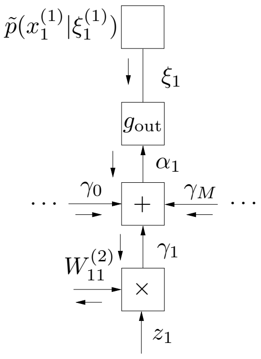


Figure E.7: Computing the derivative of the weight $W_{ij}^{(2)}$.

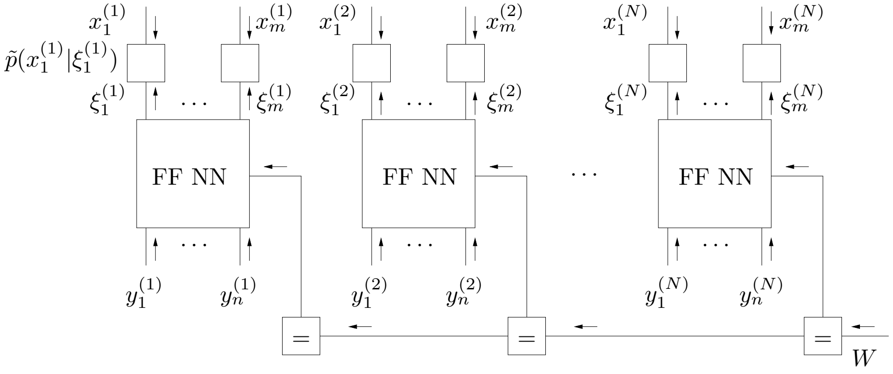


Figure E.8: Backpropagation as message passing: bottom-top sweep.

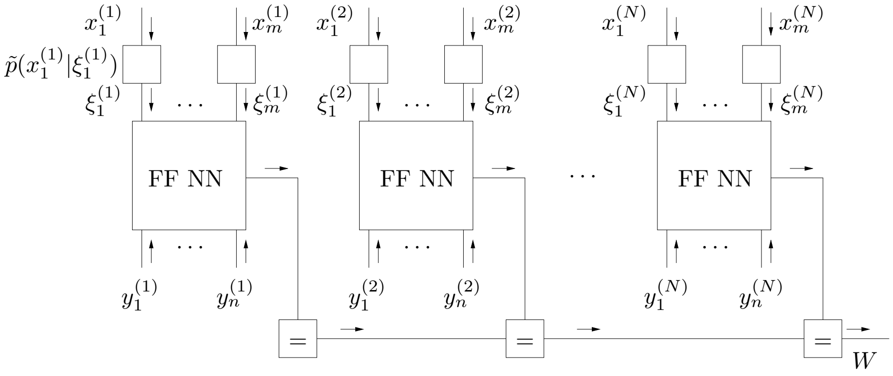


Figure E.9: Backpropagation as message passing: top-bottom sweep.

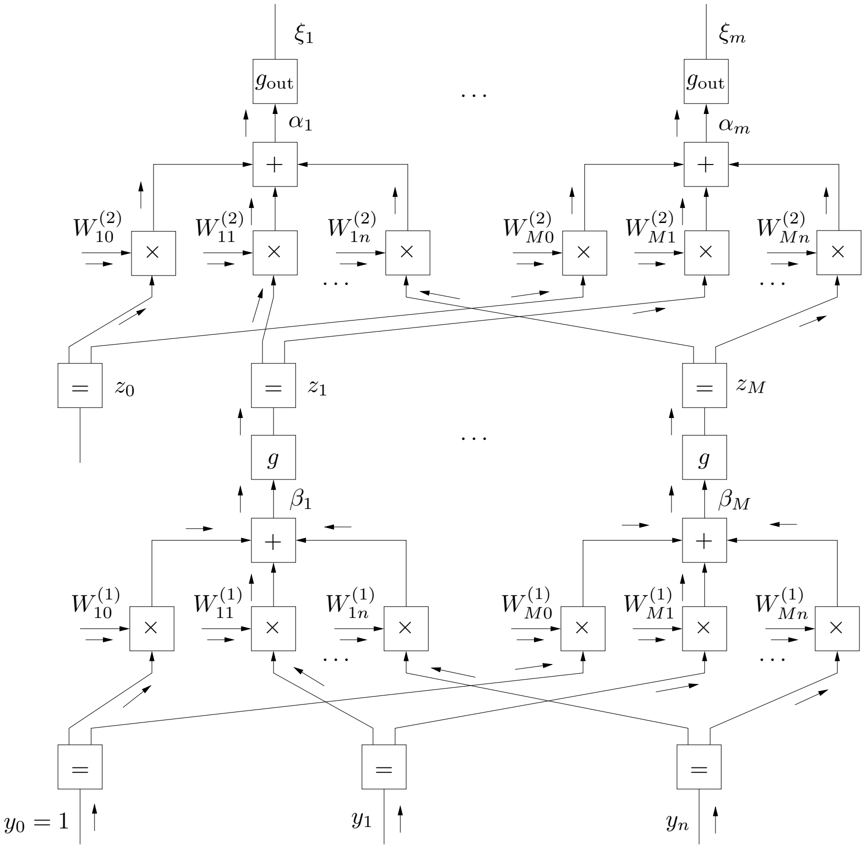


Figure E.10: Backpropagation as message passing: bottom-top sweep inside the feed-forward neural-network.

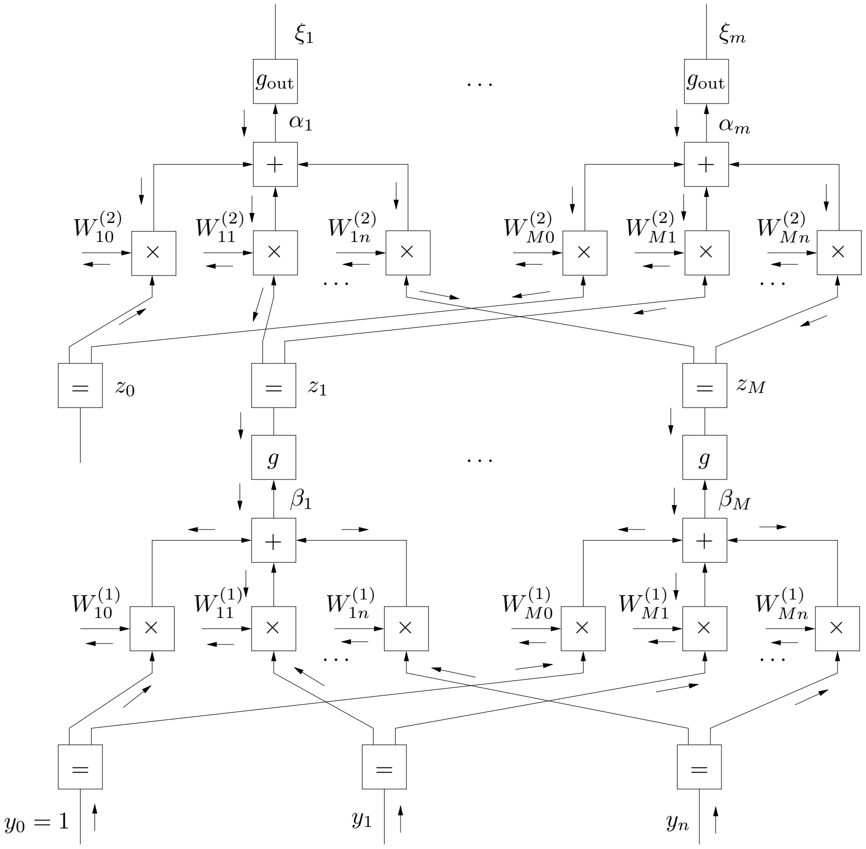


Figure E.11: Backpropagation as message passing: top-bottom sweep inside the feed-forward neural-network.

Appendix F

Some Distributions

F.1 The Gauss distribution

The Gaussian distribution $\mathcal{N}(x \mid m, v)$ (or "normal distribution") with mean (and mode) m and variance v is defined as:

$$\mathcal{N}(x \mid m, v) \triangleq \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{(x-m)^2}{2v}\right), \qquad x \in \mathbb{R}. \tag{F.1}$$

Alternatively, $\mathcal{N}(x \mid m, v)$ is denoted by $\mathcal{N}^{-1}(x \mid m, w)$, where $w = v^{-1}$ is called the precision.

In the vector case, the normal distribution is defined as:

$$\mathcal{N}^{-1}(\mathbf{x} \mid \mathbf{m}, \mathbf{W}) = \sqrt{\frac{|\mathbf{W}|}{(2\pi)^n}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^H \mathbf{W} (\mathbf{x}-\mathbf{m})\right), \quad \mathbf{x} \in \mathbb{R}^n, \tag{F.2}$$

and if $\mathbf{V} = \mathbf{W}^{-1}$ exists as:

$$\mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{V}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^H \mathbf{V}^{-1} (\mathbf{x}-\mathbf{m})\right), \quad \mathbf{x} \in \mathbb{R}^n. \tag{F.3}$$

F.2 The inverted gamma distribution

The inverted-gamma distribution $\mathrm{Ig}\,(x \mid \alpha, \beta)$ with parameters α and β $(\alpha, \beta \in \mathbb{R})$ is defined as:

$$\mathrm{Ig}\,(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-(\alpha+1)} e^{-\frac{\beta}{x}}, \qquad x > 0. \tag{F.4}$$

The mean $\mathrm{E}[X]$, variance $\mathrm{Var}[X]$ and mode $\mathrm{M}[X]$ of the distribution are:

$$\mathrm{E}[X] = \frac{\beta}{\alpha - 1} \qquad \alpha > 1 \tag{F.5}$$

$$\mathrm{Var}[X] = \frac{\beta^2}{(\alpha - 1)^2(\alpha - 2)} \qquad \alpha > 2 \tag{F.6}$$

$$\mathrm{M}[X] = \frac{\beta}{\alpha + 1}. \tag{F.7}$$

Appendix G

Gaussian Densities and Quadratic Forms

We closely follow [120] in this appendix. Multi-variable Gaussian densities are closely related to quadratic forms. A quadratic form is a function $q : \mathbb{R}^n \to \mathbb{R}$ or $\mathbb{C}^n \to \mathbb{R}$ of the form

$$q(x) = (x-m)^H W(x-m) + c \tag{G.1}$$
$$= x^H Wx - 2\mathrm{Re}\big(x^H Wm\big) + m^H Wm + c, \tag{G.2}$$

where W is a positive semi-definite $n \times n$ matrix. The case where all quantities in (G.1) are real-valued and q is a function $\mathbb{R}^n \to \mathbb{R}$ will be referred to as "the real case"; the case where q is a function $\mathbb{C}^n \to \mathbb{R}$ will be referred to as "the complex case".

A n-dimensional Gaussian distribution is a function $\mathbb{R}^n \to \mathbb{R}$ or $\mathbb{C}^n \to \mathbb{R}$ of the form

$$f(x) = \gamma e^{-q(x)}, \tag{G.3}$$

where $q(x)$ is a quadratic form as in (G.1) with positive definite W and with a scale factor γ such that $\int_{-\infty}^{\infty} f(x)\, dx = 1$.

We note without proof that, in the real case, the corresponding covariance matrix is $\frac{1}{2}W^{-1}$ and in the complex case, the covariance matrix is W^{-1}.

In the following theorem, the vector x is split into two components: $x = (y, z)$. Then (G.1) can be written as

$$q(y,z) = \Big((y-m_Y)^H,\, (z-m_Z)^H\Big) \begin{pmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{pmatrix} \begin{pmatrix} y-m_Y \\ z-m_Z \end{pmatrix}. \tag{G.4}$$

Theorem G.1. (Gaussian Max/Int Theorem)
Let $q(y,z)$ be a quadratic form as in (G.4) with $W_{1,1}$ positive definite. Then

$$\int_{-\infty}^{\infty} e^{-q(y,z)}\, dy \;\propto\; \max_y e^{-q(y,z)} \tag{G.5}$$

$$= e^{-\min_y q(y,z)}. \tag{G.6}$$

where "$\propto$" denotes equality up to a scale factor.

Proof: Let us first consider the integral over (G.1):

$$\int_{-\infty}^{\infty} e^{-q(x)}\, dx = e^{-c} \int_{-\infty}^{\infty} e^{-(q(x)-c)}\, dx \tag{G.7}$$

$$= e^{-c} \int_{-\infty}^{\infty} e^{-(x-m)^H W (x-m)}\, dx \tag{G.8}$$

$$= e^{-c} \int_{-\infty}^{\infty} e^{-x^H W x}\, dx \tag{G.9}$$

$$= e^{-\min_x q(x)} \int_{-\infty}^{\infty} e^{-x^H W x}\, dx. \tag{G.10}$$

Now consider (G.4) as a function of y with parameter z. Clearly, this function is of the form (G.2) with $W_{1,1}$ taking the role of W. It thus follows from (G.10) that

$$\int_{-\infty}^{\infty} e^{-q(y,z)}\, dy = e^{-\min_y q(y,z)} \int_{-\infty}^{\infty} e^{-y^H W_{1,1} y}\, dy. \tag{G.11}$$

But the integral on the right-hand side does not depend on z, which proves (G.6). □

Theorem G.2. (Sum of Quadratic Forms)
Let both A and B be nonnegative definite matrices (which implies that they are Hermitian). Then

$$(x-a)^H A (x-a) + (x-b)^H B (x-b) = x^H W x - 2\mathrm{Re}(x^H W m) + m^H W m + c \tag{G.12}$$

with

$$\begin{aligned} W &= A + B && \text{(G.13)} \\ m &= (A+B)^{\#}(Aa + Bb) && \text{(G.14)} \end{aligned}$$

and with the scalar

$$c = (a-b)^H A(A+B)^{\#} B(a-b). \tag{G.15}$$

We also note

$$Wm = Aa + Bb. \tag{G.16}$$

We omit the proof here.

Appendix H

Kalman Filtering and Related Topics

In this section we review several topics related to Kalman filtering and smoothing; we will closely follow [120].

H.1 Introduction to Kalman Filtering

Many problems in signal processing and control theory can be put into the following form (or some variation of it). Let $X = (X_0, X_1, X_2, \ldots)$ and $Y = (Y_1, Y_2, \ldots)$ be discrete-time stochastic processes that can be described by the linear state-space model

$$X_k = AX_{k-1} + BU_k \tag{H.1}$$

$$Y_k = CX_k + W_k. \tag{H.2}$$

The input signal $U = (U_1, U_2, \ldots)$ is a white Gaussian process (i.e., U_k is zero-mean Gaussian, independent of $U_1, U_2, \ldots, U_{k-1}$, and has the same distribution as U_1). The signal $W = (W_1, W_2, \ldots)$ is also a white Gaussian process and independent of U. All involved signals (X, Y, U, W) are real or complex vectors (e.g., X_k takes values in $\mathbb{R}^N$ or $\mathbb{C}^N$), and A, B, C are matrices of appropriate dimensions.

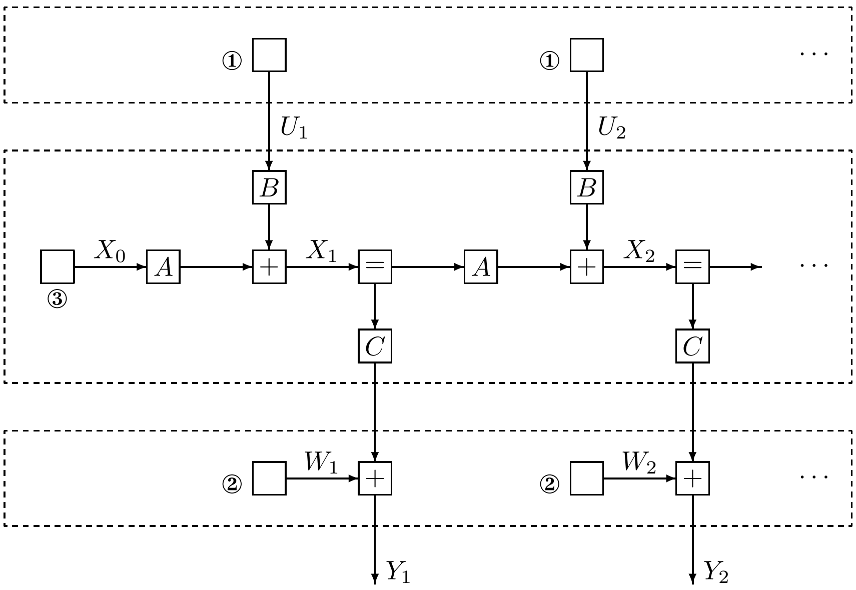


Figure H.1: Linear state-space model driven by white Gaussian noise and observed through AWGN channel.

This type of model can be described by the factor graph of Fig. H.1. The nodes labeled ① and ② represent Gaussian distributions. For example, if U_k is real-valued and scalar, then the nodes labeled ① represent the function

$$f_{①}(u_k) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-u_k^2}{2\sigma_U^2}\right). \tag{H.3}$$

Complex and vector-valued Gaussian variables will be discussed later.

The single node ③ stands for the distribution of the initial state X_0. We will assume that this is also a Gaussian distribution or else it is the constant 1 (in which case this node may be omitted from the factor graph).

The other nodes in Fig. H.1 represent deterministic relations, which give rise to factors involving Dirac deltas. For example, a matrix multiplication $Z_k = CX_k$ gives rise to the factor $\delta(z_k - Cx_k)$.

The global function that is represented by Fig. H.1 is the joint probability density of all involved variables. More precisely, for any positive integer n, the first n sections of the factor graph represent the density

$$\begin{aligned} & f(x_0, \ldots, x_n, u_1, \ldots, u_n, w_1, \ldots, w_n, y_1, \ldots, y_n) \\ &\quad = f(x_0) \prod_{k=1}^{n} f(u_k)\, f(x_k|x_{k-1}, u_k)\, f(w_k)\, f(y_k|x_k, w_k) && \text{(H.4)} \\ &\quad = f(x_0) \prod_{k=1}^{n} f(u_k)\, \delta(x_k - Ax_{k-1} - Bu_k)\, f(w_k)\, \delta(y_k - Cx_k - w_k). && \text{(H.5)} \end{aligned}$$

Suppose we have observed $(Y_1, \ldots, Y_n) = (y_1, \ldots, y_n)$ and we wish to estimate X_m. This leads to the following classical problems:

$m = n$ (filtering): estimating the current state;

$m < n$ (smoothing): estimating some past state;

$m > n$ (prediction): estimating some future state.

All these estimates can be efficiently computed (with complexity linear in n) by the Kalman filtering (or Kalman smoothing) algorithm. We will derive these algorithms as message passing in the factor graph of Fig. H.1. All messages will be Gaussian distributions—represented, e.g., by a mean vector and a covariance matrix. Since the factor graph has no cycles (and since all message computations will be exact), the sum-product algorithm will yield the correct *a posteriori* distributions. For both the filtering and the prediction problem, a single left-to-right sweep of message computations (referred to as the forward recursion) will do; for the smoothing problem, an additional right-to-left sweep (the backward recursion) is also required.

The derivation of Kalman filtering and smoothing as summary propagation in the factor graph makes it easy to generalize both the problem and the estimation algorithms in many ways and to adapt it to many applications. Such applications include, e.g., various parameter estimation tasks in a communications receiver; the factor graph approach makes it easy to integrate such estimation algorithms together with the error correcting decoder into a coherent iterative message passing receiver architecture.

If all variables in Fig. H.1 are real scalars (which implies that all matrices in Fig. H.1 are scalars as well), all messages will be one-dimensional Gaussian distributions. In other words, all messages are functions of the form

$$\mu(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(\xi - m)^2}{2\sigma^2} \right) \tag{H.6}$$

and are fully described by their mean m and their variance σ^2. The rules for the computation of the messages are tabulated in Table H.1. As only one of the two messages along any edge (say X) is considered, the corresponding means and variances are simply denoted m_X, σ_X, etc.

The proofs of the message computation rules of Table H.1 are not difficult. Rule 3 amounts to the well-known fact that the mean of aX is am_X and the variance of aX is a^2 times the variance of X. Rule 4 is (in the scalar case) equivalent to Rule 3. Rule 2 amounts to the well-known fact that the mean of $X + Y$ is $m_X + m_Y$ and (assuming that X and Y are independent) the variance of $X + Y$ is the sum of the variances. Note that these three rules hold for the mean and the variance of arbitrary distributions; the Gaussian assumption is not needed for these rules. However, Gaussian input messages clearly lead to Gaussian output messages.

Rule 1 of Table H.1 requires more work. Applying the sum-product rule with integration instead of summation (and with a scale factor γ yields

$$\begin{aligned} \mu_Z(z) &= \gamma \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(z - x)\, \delta(z - y)\, \mu_X(x)\, \mu_Y(y)\, dx\, dy && \text{(H.7)} \\ &= \gamma\, \mu_X(z)\, \mu_Y(z) && \text{(H.8)} \\ &= \gamma \frac{1}{\sqrt{2\pi}\sigma_X} \exp\left(\frac{-(z - m_X)^2}{2\sigma_X^2} \right) \frac{1}{\sqrt{2\pi}\sigma_Y} \exp\left(\frac{-(z - m_Y)^2}{2\sigma_Y^2} \right). && \text{(H.9)} \end{aligned}$$

After some computations, this leads to

$$\mu_Z(z) = \frac{1}{\sqrt{2\pi}\sigma_Z} \exp\left(\frac{-(z - m_Z)^2}{2\sigma_Z^2} \right) \tag{H.10}$$

with m_Z and σ_Z^2 as in Table H.1 and where γ was chosen such that $\int_{-\infty}^{\infty} \mu_Z(z) = 1$. (The details are left as an exercise.)

A remarkable property of Table H.1 is that all variances are computed only from other variances; the means are not used. In the standard setup

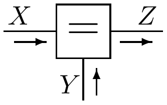
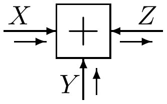
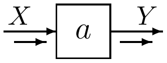
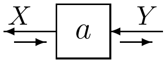
1	 $\delta(x-y)\delta(x-z)$	$m_Z = \dfrac{m_X/\sigma_X^2 + m_Y/\sigma_Y^2}{1/\sigma_X^2 + 1/\sigma_Y^2}$ $1/\sigma_Z^2 = 1/\sigma_X^2 + 1/\sigma_Y^2$
2	 $\delta(x+y+z)$	$m_Z = -m_X - m_Y$ $\sigma_Z^2 = \sigma_X^2 + \sigma_Y^2$
3	 $\delta(y-ax)$	$m_Y = am_X$ $\sigma_Y^2 = a^2\sigma_X^2$
4	 $\delta(x-ay)$	$m_Y = m_X/a$ $\sigma_Y^2 = \sigma_X^2/a^2$

Table H.1: Computation of scalar Gaussian messages consisting of mean m and variance σ^2.

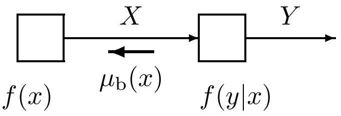


Figure H.2: Backward message in chain rule model.

of Fig. H.1, this means that the variances can be computed off-line, prior to any observation.

It is also worth pointing out that, in any chain rule model as in Fig. H.2, the backward message $\mu_{\mathrm{b}}(x)$ is neutral as long as the variable Y is not observed:

$$\mu_{\mathrm{b}}(x) = \int_{-\infty}^{\infty} f(y|x)\, dy \tag{H.11}$$

$$= 1. \tag{H.12}$$

This applies, in particular, to the backwards (right-to-left) messages from the "unobserved" future in Fig. H.1. (Such neutral messages may be considered as limits of Gaussian distributions with variance $\sigma^2 \to \infty$.)

Unfortunately, the scalar Kalman filter is of little practical use. While both the input U_k and the output Y_k are sometimes (i.e., in some applications) scalars, the state X_k is almost always a vector. The resulting complications will be considered in the following section.

H.2 Kalman Filtering: Vector Case

At this point, the reader should familiarize himself with the material in Appendix G.

In particular, we point out that Theorem G.1 has a number of remarkable consequences:

- For Gaussian distributions, eliminating variables by marginalization coincides with eliminating them by maximization. This implies, in particular, that the sum-product (integral-product) message computation rule coincides with the max-product rule.

- Minimizing a quadratic cost function (say $q(y, z)$) over some of the variables (say y) is equivalent to marginalizing the corresponding Gaussian distribution.
- Successive elimination of variables by sum-product message passing in a Gaussian factor graph is equivalent to solving a quadratic minimization problem by a series of successive minimizations. In particular, Kalman filtering may be viewed as a general recursive least-squares algorithm.

The vector form of the message computation rules of Table H.1 are given in Table H.2. (The proofs are given in [117].) All messages are assumed to be of the form (G.3); they are represented by the mean vector m and either the "cost" matrix $\mathbf{W}$ (or "potential" matrix) or the covariance matrix $\mathbf{V} = \mathbf{W}^{-1}$. Note that the rules in Table H.1 can often be simplified if the involved matrices are invertible.

In general, the matrices $\mathbf{W}$ and $\mathbf{V}$ are only required to be positive semi-definite, which allows to express certainty in $\mathbf{V}$ and complete ignorance in $\mathbf{W}$. However, whenever such a matrix needs to be inverted, it had better be positive definite.

The direct application of the update rules in Table H.2 may lead to frequent matrix inversions. A key observation in Kalman filtering is that the inversion of large matrices can often be avoided. In the factor graph, such simplifications may be achieved by using the update rules for the composite blocks given in Table H.3. In particular, the vectors U_k and $\mathbf{W}_k$ in Fig. H.1 have usually much smaller dimensions than the state vector X_k; in fact, they are often scalars. By working with composite blocks as in Fig. H.3, the forward recursion (left in Fig. H.3) using the covariance matrix $\mathbf{V} = \mathbf{W}^{-1}$ then requires no inversion of a large matrix and the backward recursion (right in Fig. H.3) using the cost matrix $\mathbf{W}$ requires only one such inversion for each discrete time index.

Strictly speaking, the term "Kalman filtering" refers only to the forward (left-to-right) recursion through Fig. H.1 as in Fig. H.3 (left). In this context, the quantity

$$\mathbf{G}_{\text{Kalman}} \triangleq \mathbf{V}_X \mathbf{A}^H \mathbf{G} \tag{H.13}$$

$$= \mathbf{V}_X \mathbf{A}^H (\mathbf{V}_Y + \mathbf{A}\mathbf{V}_X \mathbf{A}^H)^{-1} \tag{H.14}$$

which appears in Rule 5 of Table H.3 is traditionally called *Kalman gain.*

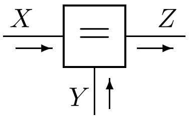
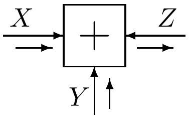
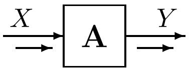
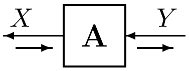
1	 $\delta(x-y)\delta(x-z)$	$m_Z = (\mathbf{W}_X + \mathbf{W}_Y)^{\#}(\mathbf{W}_X m_X + \mathbf{W}_Y m_Y)$ $\mathbf{V}_Z = \mathbf{V}_X(\mathbf{V}_X + \mathbf{V}_Y)^{\#}\mathbf{V}_Y$ $\mathbf{W}_Z = \mathbf{W}_X + \mathbf{W}_Y$
2	 $\delta(x+y+z)$	$m_Z = -m_X - m_Y$ $\mathbf{V}_Z = \mathbf{V}_X + \mathbf{V}_Y$ $\mathbf{W}_Z = \mathbf{W}_X\big(\mathbf{W}_X + \mathbf{W}_Y\big)^{\#}\,\mathbf{W}_Y$
3	 $\delta(y-\mathbf{A}x)$	$m_Y = \mathbf{A}m_X$ $\mathbf{V}_Y = \mathbf{A}\mathbf{V}_X\mathbf{A}^H$ $\mathbf{W}_Y \overset{1}{=} \mathbf{A}^{-H}\mathbf{W}_X\mathbf{A}^{-1}$
4	 $\delta(x-\mathbf{A}y)$	$m_Y = \big(\mathbf{A}^H\mathbf{W}_X\mathbf{A}\big)^{\#}\mathbf{A}^H\mathbf{W}_X m_X$ $\mathbf{V}_Y \overset{1}{=} \mathbf{A}^{-1}\mathbf{V}_X\mathbf{A}^{-H}$ $\mathbf{W}_Y = \mathbf{A}^H\mathbf{W}_X\mathbf{A}$
[1] if $\mathbf{A}$ is invertible		

Table H.2: Computation of multi-dimensional Gaussian messages consisting of mean vector m and covariance matrix $\mathbf{V}$ or $\mathbf{W} = \mathbf{V}^{-1}$. Notation: $(.)^H$ denotes Hermitian transposition and $(.)^{\#}$ denotes the Moore-Penrose pseudo-inverse.

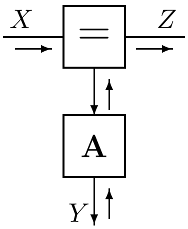
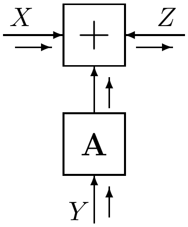
5		$m_Z = m_X + \mathbf{V}_X \mathbf{A}^H \mathbf{G} \left(m_Y - \mathbf{A} m_X \right)$ $\mathbf{V}_Z = \mathbf{V}_X - \mathbf{V}_X \mathbf{A}^H \mathbf{G} \mathbf{A} \mathbf{V}_X$ $\mathbf{W}_Z = \mathbf{W}_X + \mathbf{A}^H \mathbf{W}_Y \mathbf{A}$ with $\mathbf{G} \triangleq \left(\mathbf{V}_Y + \mathbf{A} \mathbf{V}_X \mathbf{A}^H \right)^{-1}$
6		$m_Z = -m_X - \mathbf{A} m_Y$ $\mathbf{V}_X \overset{1}{=} \mathbf{A}^{-1} \mathbf{V}_Y \mathbf{A}^{-H}$ $\mathbf{W}_Z = \mathbf{W}_X - \mathbf{W}_X \mathbf{A} \mathbf{H} \mathbf{A}^H \mathbf{W}_X$ with $\mathbf{H} \triangleq \left(\mathbf{W}_Y + \mathbf{A}^H \mathbf{W}_X \mathbf{A} \right)^{-1}$
[1] if $\mathbf{A}$ is invertible		

Table H.3: Update rules for composite blocks.

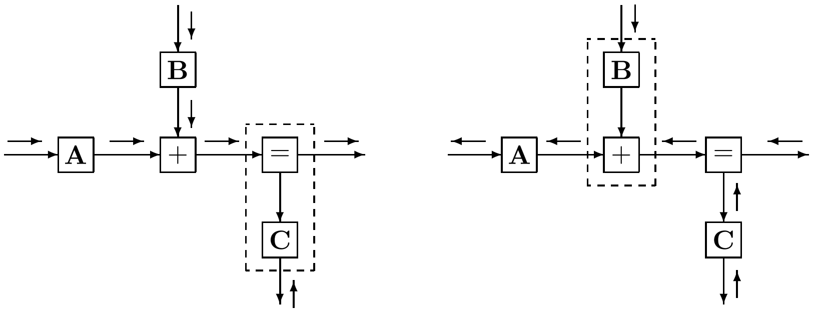


Figure H.3: Use of the composite-block rules of Table H.3.

Appendix I

Differentiation under the Integral Sign

Theorem I.1. Suppose $f(x, y)$ and $\frac{\partial f(x,y)}{\partial x}$ are defined and continuous for all $x \in [a, b]$ and $y \in [c, d]$. Let

$$g(x) = \int_c^d f(x, y) dy. \tag{I.1}$$

Then g is differentiable and

$$\frac{dg(x)}{dx} = \int_c^d \frac{\partial f(x, y)}{\partial x} dy. \tag{I.2}$$

Proof: We refer to [106, p. 276]. □

Theorem I.2. Suppose $f(x, y)$ and $\frac{\partial f(x,y)}{\partial x}$ are defined and continuous for all $x \in [a, b]$ and $y \in [c, d]$. Let the functions $u_0(x)$ and $u_1(x)$ and their first derivatives be continuous for $x \in [a, b]$ with the range of u_0 and u_1 in (c, d). Let

$$g(x) = \int_{u_0(x)}^{u_1(x)} f(x, y) dy. \tag{I.3}$$

Then g is differentiable and

$$\frac{dg(x)}{dx} = f(x, u_1(x))\frac{du_1(x)}{dx} - f(x, u_0(x))\frac{du_0(x)}{dx} + \int_{u_0(x)}^{u_1(x)} \frac{\partial f(x,y)}{\partial x} dy. \tag{I.4}$$

Proof: We refer to [166, p. 426]. □

Theorem I.3. Suppose $f(x, y)$ and $\frac{\partial f(x,y)}{\partial x}$ are defined and continuous for all $x \in [a, b]$ and $y \geq c$. Assume that there are functions $\phi(x)$ and $\psi(x)$ which are ≥ 0 for all $x \in [a, b]$, such that $|f(x, y)| \leq \phi(y)$ and $|\frac{\partial f(x,y)}{\partial x}| \leq \psi(y)$ for all x and y, and such that the integrals

$$\int_c^\infty \phi(y)dy \qquad \text{and} \qquad \int_c^\infty \psi(y)dy \tag{I.5}$$

converge.

Let

$$g(x) = \int_c^\infty f(x, y)dy. \tag{I.6}$$

Then g is differentiable and

$$\frac{dg(x)}{dx} = \int_c^\infty \frac{\partial f(x,y)}{\partial x} dy. \tag{I.7}$$

Proof: We refer to [106, pp. 337–339]. □

Theorem I.4. Suppose $f(x, y)$ and $\frac{\partial f(x,y)}{\partial x}$ are defined and continuous for all $x \in [a, b]$ and $y \geq c$. Assume that

$$\int_c^\infty \frac{\partial f(x,y)}{\partial x} dy \tag{I.8}$$

converges uniformly for all $x \in [a, b]$, and that

$$g(x) = \int_c^\infty f(x, y)dy \tag{I.9}$$

converges for all $x \in [a, b]$.

Then g is differentiable and

$$\frac{dg(x)}{dx} = \int_c^\infty \frac{\partial f(x,y)}{\partial x} dy. \tag{I.10}$$

Proof: We refer to [106, p. 340]. □

Appendix J

Derivation of the EM Update Rules

J.1 Mean Estimation

J.1.1 The Scalar Case

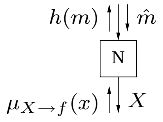


Figure J.1: Factor graph node for a Gaussian distribution with unknown mean.

We consider the situation depicted in Fig. J.1. The node in Fig. J.1 represents the function:

$$f(x,m) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s}\right), \tag{J.1}$$

where $m \in \mathbb{R}$ is the unknown mean and $s \in \mathbb{R}^+$ the known variance of the Gaussian distribution. The h-message is computed as follows:

$$h(m) = \int_x p(x|\hat{m}^{(k)}) \log f(x,m) dx \tag{J.2}$$

$$= \int_x p(x|\hat{m}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx \tag{J.3}$$

$$= C - \frac{1}{2s}\left(m^2 - 2m\mathrm{E}[X|\hat{m}^{(k)}] \right), \tag{J.4}$$

where C is a proper scaling constant. As a consequence,

$$\boxed{e^{h(m)} \propto \mathcal{N}\left(m \,\middle|\, \mathrm{E}[X|\hat{m}^{(k)}], s \right)} \tag{J.5}$$

J.1.2 The Vector Case

In the vector case, the node function equals:

$$f(\mathbf{x},\mathbf{m}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^H \mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) \right). \tag{J.6}$$

The h-message is given by:

$$h(\mathbf{m}) = \int_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{m}}^{(k)}) \log f(\mathbf{x},\mathbf{m}) d\mathbf{x} \tag{J.7}$$

$$= C - \frac{1}{2}\left(\mathbf{m}^H \mathbf{V}^{-1}\mathbf{m} - 2\mathbf{m}^H \mathbf{V}^{-1}\mathrm{E}[\mathbf{X}|\hat{\mathbf{m}}^{(k)}] \right). \tag{J.8}$$

As a consequence,

$$\boxed{e^{h(\mathbf{m})} \propto \mathcal{N}\left(\mathbf{m} \,\middle|\, \mathrm{E}[\mathbf{X}|\hat{\mathbf{m}}^{(k)}], \mathbf{V} \right)} \tag{J.9}$$

J.2 Variance Estimation

J.2.1 The Scalar Case

We consider the situation depicted in Fig. J.2. The node in Fig. J.2 represents the function

$$f(x,s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x-m)^2}{2s} \right), \tag{J.10}$$

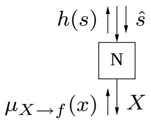


Figure J.2: Factor graph node for a Gaussian distribution with unknown variance.

where $s \in \mathbb{R}^+$ is the unknown variance and $m \in \mathbb{R}$ the known mean of the Gaussian distribution. The h-message is computed as follows:

$$\begin{aligned} h(s) &= \int_x p(x|\hat{s}^{(k)}) \log f(x,s)dx && \text{(J.11)} \\ &= \int_x p(x|\hat{s}^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x-m)^2}{2s} \right) dx && \text{(J.12)} \\ &= C - \frac{1}{2}\log s - \frac{\mathrm{E}\left[(X-m)^2 \mid \hat{s}^{(k)}\right]}{2s}. && \text{(J.13)} \end{aligned}$$

Therefore,

$$\boxed{e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{1}{2}\mathrm{E}\left[(X-m)^2 \mid \hat{s}^{(k)}\right] \right)} \quad \text{(J.14)}$$

where Ig denotes an inverted gamma distribution.

J.2.2 The Vector Case

In the vector case, the node function equals:

$$f(\mathbf{x},\mathbf{V}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^H \mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) \right). \quad \text{(J.15)}$$

The h-message is therefore:

$$\begin{aligned} h(\mathbf{V}) &= \int_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{V}}^{(k)}) \log f(\mathbf{x},\mathbf{V}) d\mathbf{x} && \text{(J.16)} \\ &= C - \frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\mathrm{E}\left[(\mathbf{X}-\mathbf{m})^H \mathbf{V}^{-1}(\mathbf{X}-\mathbf{m}) \,\middle|\, \hat{\mathbf{V}}^{(k)} \right]. && \text{(J.17)} \end{aligned}$$

J.2.3 Special Forms of V

$\mathbf{V} = \mathbf{I}s, s \in \mathbb{R}^+$:

$$h(s) = C - \frac{n}{2}\log s - \frac{1}{2s}\mathrm{E}\left[(\mathbf{X}-\mathbf{m})^H(\mathbf{X}-\mathbf{m})\right] \tag{J.18}$$

$$e^{h(s)} = \mathrm{Ig}\left(s \,\middle|\, \frac{n-2}{2}, \frac{1}{2}\mathrm{E}\left[(\mathbf{X}-\mathbf{m})^H(\mathbf{X}-\mathbf{m})\right]\right). \tag{J.19}$$

$\mathbf{V} = \mathrm{diag}\left(\mathbf{s}\right), \mathbf{s} \in \mathbb{R}^{+n}$:

$$h(\mathbf{s}) = C - \frac{1}{2}\sum_{\ell=1}^{n}\log s_\ell - \sum_{\ell=1}^{n}\frac{1}{2s_\ell}\mathrm{E}[(X_\ell - m_\ell)^2] \tag{J.20}$$

$$\propto \prod_{\ell=1}^{n}\mathrm{Ig}\left(s_\ell \,\middle|\, -\frac{1}{2}, \frac{1}{2}\mathrm{E}[(X_\ell - m_\ell)^2]\right). \tag{J.21}$$

J.3 Coefficient Estimation

J.3.1 The Scalar Case

Here, the problem of estimating the coefficients of an autoregressive (AR) system is considered. The function $f(x_1, x_2, a)$ is defined as:

$$f(x_1, x_2, a) = \delta(x_2 - a x_1), \tag{J.22}$$

where $a \in \mathbb{R}$ is the scalar AR coefficient. Computing the message $h(a)$ for this node leads to a singularity problem because of the Dirac delta (J.22). We can avoid this problem by combining the Dirac delta with a *continuous* neighboring node. In the AR model, there is also a driving input (or control input) as depicted in Fig. J.3. The dotted box in the graph to the left is represented by the node to the right.

The function represented by the graph in Fig. J.3 is

$$f(x_1, x_2, a) = \frac{1}{\sqrt{2\pi s}}\exp\left(-\frac{(x_2 - a x_1)^2}{2s}\right). \tag{J.23}$$

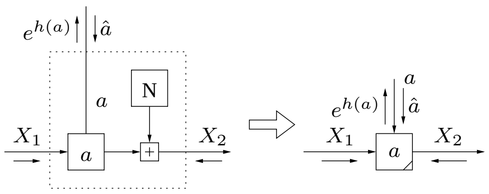


Figure J.3: Factor graph of the state transition node for a linear state-space model.

The message $h(a)$ becomes:

$$h(a) = \int_{x_1,x_2} p(x_1, x_2|a^{(k)}) \log f(x_1, x_2, a) dx_1 dx_2 \tag{J.24}$$

$$= \int_{x_1,x_2} p(x_1, x_2|a^{(k)}) \left(-\frac{1}{2}\log(2\pi s) - \frac{(x_2 - ax_1)^2}{2s}\right) dx_1 dx_2 \tag{J.25}$$

$$= C - \frac{1}{2s}\left(a^2 \mathrm{E}[X_1^2|\hat{a}^{(k)}] - a2\mathrm{E}[X_1X_2|\hat{a}^{(k)}] + \mathrm{E}[X_2^2|\hat{a}^{(k)}]\right), \tag{J.26}$$

with

$$C = -\frac{1}{2}\log(2\pi s). \tag{J.27}$$

Because (J.26) is a quadratic form, it is convenient to send the message $e^{h(a)}$ instead of $h(a)$:

$$\boxed{e^{h(a)} \propto \mathcal{N}^{-1}\left(a \,\middle|\, \frac{\mathrm{E}[X_1X_2|\hat{a}^{(k)}]}{\mathrm{E}[X_1^2|\hat{a}^{(k)}]}, \mathrm{E}[X_1^2|\hat{a}^{(k)}]s^{-1}\right)} \tag{J.28}$$

J.3.2 The Vector Case

The function $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{A})$ is defined as:

$$f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{A}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{V}|}} \exp\left(-\frac{1}{2}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^H \mathbf{V}^{-1}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)\right), \tag{J.29}$$

with $\mathbf{A} \in \mathbb{R}^{n\times n}$. The message $h(\mathbf{A})$ becomes:

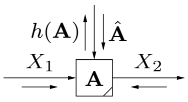


Figure J.4: Factor graph of the state transition node for the vector case.

$$\begin{aligned}
h(\mathbf{A}) &= \int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1,\mathbf{x}_2|\hat{\mathbf{A}}^{(k)}) \log f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{A}) d\mathbf{x}_1 d\mathbf{x}_2 && \text{(J.30)}\\
&= C - \frac{1}{2}\int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1,\mathbf{x}_2|\hat{\mathbf{A}}^{(k)})\\
&\qquad (\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)^H\mathbf{V}^{-1}(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)d\mathbf{x}_1 d\mathbf{x}_2 && \text{(J.31)}\\
&= C - \frac{1}{2}\Big(\mathrm{E}[\mathbf{X}_2^H\mathbf{V}^{-1}\mathbf{X}_2|\hat{\mathbf{A}}^{(k)}] - 2\mathrm{E}[\mathbf{X}_2^H\mathbf{V}^{-1}\mathbf{A}\mathbf{X}_1|\hat{\mathbf{A}}^{(k)}]\\
&\qquad +\mathrm{E}[\mathbf{X}_1^H\mathbf{A}^H\mathbf{V}^{-1}\mathbf{A}\mathbf{X}_1|\hat{\mathbf{A}}^{(k)}]\Big) && \text{(J.32)}\\
&= C - \frac{1}{2}\mathrm{E}\left[\|\mathbf{X}_2-\mathbf{A}\mathbf{X}_1\|^2_{\mathbf{V}^{-1}}\middle|\hat{\mathbf{A}}^{(k)}\right], && \text{(J.33)}
\end{aligned}$$

with

$$C = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{V}|. \tag{J.34}$$

Unfortunately, (J.33) does not have a nice form in $\mathbf{A}$. Unless a special structure is imposed onto $\mathbf{A}$, it is impossible to parameterise (J.33).

J.3.3 The AR Case

Here, a special case of Section J.3.2 is treated. The function $f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a})$ is defined as

$$f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{V}|}}\exp\left(-\frac{1}{2}(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)^H\mathbf{V}^{-1}(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1)\right), \tag{J.35}$$

with

$$\mathbf{A} \triangleq \left[\begin{array}{cc} \multicolumn{2}{c}{\mathbf{a}^H} \\ \mathbf{I} & \mathbf{0}\end{array}\right]. \tag{J.36}$$

The message $h(\mathbf{a})$ becomes

$$\begin{aligned} h(\mathbf{a}) &= \int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1,\mathbf{x}_2|\mathbf{a}^{(k)}) \log f(\mathbf{x}_1,\mathbf{x}_2,\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2 & \text{(J.37)} \\ &= C - \frac{1}{2}\int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1,\mathbf{x}_2|\mathbf{a}^{(k)}) \\ &\qquad (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a})^H \mathbf{V}^{-1}(\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2, & \text{(J.38)} \end{aligned}$$

with

$$C = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{V}|, \tag{J.39}$$

and where

$$\mathbf{S} \triangleq \begin{bmatrix} & \mathbf{0}^H \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \qquad \mathbf{c} \triangleq (1,0,\ldots,0)^H. \tag{J.40}$$

The product $\mathbf{A}\mathbf{x}_1$ is separated in the shifting operation $\mathbf{S}\mathbf{x}_1$, which shifts every element in the vector $\mathbf{x}_1$ one position down, and the inner vector product $\mathbf{c}\mathbf{x}_1^H\mathbf{a}$.

We write the RHS of (J.38) as a quadratic form in $\mathbf{a}$:

$$\begin{aligned} h(\mathbf{a}) &= C - \frac{1}{2}\int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1,\mathbf{x}_2|\mathbf{a}^{(k)}) \\ &\qquad (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a})^H \mathbf{V}^{-1}(\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2, & \text{(J.41)} \\ &\triangleq C - \frac{1}{2}\mathrm{E}\left[(\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a})^H \mathbf{V}^{-1}(\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a})\right] & \text{(J.42)} \\ &= C' - \frac{1}{2}\mathrm{E}\left[-\mathbf{m}_a^H\mathbf{W}_a\,\mathbf{a} - \mathbf{a}^H\mathbf{W}_a\,\mathbf{m}_a + \mathbf{a}^H\mathbf{W}_a\,\mathbf{a}\right], & \text{(J.43)} \end{aligned}$$

where C' is an irrelevant constant and

$$\begin{aligned} \mathbf{W}_a &= \mathrm{E}\left[\mathbf{X}_1\mathbf{c}^H\mathbf{V}^{-1}\mathbf{c}\mathbf{X}_1^H\right] & \text{(J.44)} \\ \mathbf{m}_a &= \mathbf{W}_a^{-1}\mathrm{E}\left[\mathbf{X}_1\mathbf{c}^H\mathbf{V}^{-1}(\mathbf{X}_2 - \mathbf{S}\mathbf{X}_1)\right]. & \text{(J.45)} \end{aligned}$$

Note that $\mathbf{c}^H\mathbf{V}^{-1}\mathbf{c}$ is nothing but the element $\left[\mathbf{V}^{-1}\right]_{11}$, i.e., the element (1,1) of the matrix $\mathbf{V}^{-1}$. Therefore, we can rewrite (J.44) as

$$\mathbf{W}_a = \left[\mathbf{V}^{-1}\right]_{11}\mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right]. \tag{J.46}$$

Note also that $\mathbf{c}^H\mathbf{V}^{-1}$ is the first row of $\mathbf{W} \triangleq \mathbf{V}^{-1}$, i.e.,

$$\mathbf{c}^H\mathbf{V}^{-1} = (w_{11}, w_{12}, \ldots, w_{1n}), \tag{J.47}$$

where w_{ij} is the element (i,j) of $\mathbf{W}$ $(i,j = 1,\ldots,n)$. On the other hand, it is easy to verify that

$$\mathbf{c}^H\mathbf{V}^{-1}\mathbf{S} = (w_{12},\ldots,w_{1n},0). \tag{J.48}$$

By substituting (J.47) and (J.48) in (J.45), we obtain:

$$\begin{aligned}
\mathbf{m}_a &= \mathbf{W}_a^{-1}\mathrm{E}\left[\mathbf{X}_1\mathbf{c}^H\mathbf{V}^{-1}(\mathbf{X}_2-\mathbf{S}\mathbf{X}_1)\right] && \text{(J.49)}\\
&= \mathbf{W}_a^{-1}\left(\mathrm{E}\left[\mathbf{X}_1\mathbf{c}^H\mathbf{V}^{-1}\mathbf{X}_2\right]-\mathrm{E}\left[\mathbf{X}_1\mathbf{c}^H\mathbf{V}^{-1}\mathbf{S}\mathbf{X}_1\right]\right) && \text{(J.50)}\\
&= \mathbf{W}_a^{-1}\left(\sum_{k=1}^{n} w_{1,k}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_2\right]_k\right]-\sum_{k=1}^{n-1} w_{1,k+1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_1\right]_k\right]\right), && \text{(J.51)}
\end{aligned}$$

where $[\mathbf{X}_i]_j$ is the j-th component of the (random) vector $\mathbf{X}_i$ $(i = 1,2; j = 1,\ldots,n)$.

In this case, the message $e^{h(\mathbf{a})}$ has the parametrization:

$$e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\left(\mathbf{a}\,\middle|\,\mathbf{m}_a,\mathbf{W}_a\right), \tag{J.52}$$

where $\mathbf{m}_a$ and $\mathbf{W}_a$ is given by (J.51) and (J.46) respectively.

Special cases

$\mathbf{V} = \mathrm{diag}\,(\mathbf{s})\,,\mathbf{s} \triangleq (s_1,\ldots,s_n) \in \mathbb{R}^{+n}$:

$$\begin{aligned}
\mathbf{W}_a &= \frac{1}{s_1}\mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right] && \text{(J.53)}\\
\mathbf{m}_a &= \mathbf{W}_a^{-1}\frac{1}{s_1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_2\right]_1\right] && \text{(J.54)}\\
&= \mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right]^{-1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_2\right]_1\right]. && \text{(J.55)}
\end{aligned}$$

$\mathbf{V} = \mathbf{I}s, s \in \mathbb{R}^{+}$:

$$\begin{aligned}
\mathbf{W}_a &= \frac{1}{s}\mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right] && \text{(J.56)}\\
\mathbf{m}_a &= \mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right]^{-1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_2\right]_1\right]. && \text{(J.57)}
\end{aligned}$$

J.4 Joint Coefficient and Variance Estimation

J.4.1 The Scalar Case

Here, the problem of jointly estimating the coefficients and the variance of the driving noise of an autoregressive (AR) system is considered. The

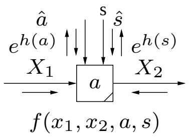


Figure J.5: Factor graph of the state transition node for joint coefficient/variance estimation.

node function $f(x_1, x_2, a, s)$ is defined as

$$f(x_1, x_2, a, s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x_2 - ax_1)^2}{2s}\right), \tag{J.58}$$

with $a \in \mathbb{R}$ the scalar AR coefficient.

The message $h(a, s)$ becomes

$$\begin{aligned} h(a, s) &= \int_{x_1,x_2} p(x_1, x_2|\hat{a}^{(k)}, \hat{s}^{(k)}) \log f(x_1, x_2, a, s) dx_1 dx_2 && \text{(J.59)} \\ &= \int_{x_1,x_2} p(x_1, x_2|\hat{a}^{(k)}, \hat{s}^{(k)}) \\ &\qquad \left(-\frac{1}{2}\log(2\pi s) - \frac{(x_2 - ax_1)^2}{2s}\right) dx_1 dx_2 && \text{(J.60)} \\ &= C - \frac{1}{2}\log s - \frac{1}{2s}\left(a^2 \mathrm{E}\left[X_1^2|\hat{a}^{(k)}, \hat{s}^{(k)}\right]\right. \\ &\qquad \left. -a2\mathrm{E}\left[X_1 X_2|\hat{a}^{(k)}, \hat{s}^{(k)}\right] + \mathrm{E}\left[X_2^2|\hat{a}^{(k)}, \hat{s}^{(k)}\right]\right), && \text{(J.61)} \end{aligned}$$

with

$$C = -\frac{1}{2}\log(2\pi). \tag{J.62}$$

The message (J.61) is both a function of a and s. To find the maximum, all partial derivations of $h(a,s)$ are set to zero. From

$$\begin{aligned}
\frac{\partial h(a,s)}{\partial a} &= \int_{x_1,x_2} p(x_1,x_2|\hat{a}^{(k)},\hat{s}^{(k)}) \left(\frac{x_1(x_2-ax_1)}{s}\right) dx_1 dx_2 & \text{(J.63)}\\
&\overset{!}{=} 0 & \text{(J.64)}\\
\frac{\partial h(a,s)}{\partial s} &= \int_{x_1,x_2} p(x_1,x_2|\hat{a}^{(k)},\hat{s}^{(k)}) \left(-\frac{1}{2s}+\frac{(x_2-ax_1)^2}{2s^2}\right) dx_1 dx_2 & \text{(J.65)}\\
&\overset{!}{=} 0, & \text{(J.66)}
\end{aligned}$$

it follows that

$$\begin{aligned}
\hat{a}^{(k+1)} &= \frac{\mathrm{E}\left[X_1X_2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\mathrm{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]} & \text{(J.67)}\\
\hat{s}^{(k+1)} &= \mathrm{E}\left[(X_2-\hat{a}^{(k+1)}X_1)^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]. & \text{(J.68)}
\end{aligned}$$

As can be seen from (J.67) and (J.68), in certain situations, the estimation of a and s is decoupled. We therefore may send the following messages separately:

$$\boxed{\begin{aligned}
e^{h(a)} &\propto \mathcal{N}^{-1}\left(a \,\middle|\, \frac{\mathrm{E}\left[X_1X_2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\mathrm{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}, \frac{\mathrm{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\hat{s}^{(k)}}\right)\\
e^{h(s)} &\propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{\mathrm{E}\left[(X_2-\hat{a}^{(k+1)}X_1)^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{2}\right)
\end{aligned}}$$

In certain situations, the estimation of a and s is coupled, but one may still send the messages $e^{h(a)}$ and $e^{h(s)}$ of (J.4.1). This corresponds to approximating the M-step by ICM (cf. Section 4.9.5).

If one wishes to perform the exact M-step, one needs to send the (exact) message $e^{h(a,s)}$ defined as:

$$\boxed{e^{h(a,s)} \propto \mathcal{N}^{-1}\left(a \,\middle|\, \frac{\mathrm{E}\left[X_1X_2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{\mathrm{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}, \frac{\mathrm{E}\left[X_1^2|\hat{a}^{(k)},\hat{s}^{(k)}\right]}{s}\right)}$$

J.4.2 The AR Case

The node function $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}, s)$ is defined as

$$f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}, s) = \frac{1}{\sqrt{(2\pi)^2 s}} \exp\left(-\frac{(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)^H(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1)}{2s}\right), \tag{J.69}$$

with

$$\mathbf{A} \triangleq \left[\begin{array}{c} \mathbf{a}^H \\ \mathbf{I} \quad \mathbf{0} \end{array}\right] \tag{J.70}$$

and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{V} = \mathbf{I}s$.

The message $h(\mathbf{a}, s)$ becomes:

$$h(\mathbf{a}, s) = \int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)} \hat{s}^{(k)}) \log f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{a}, s) d\mathbf{x}_1 d\mathbf{x}_2 \tag{J.71}$$

$$= C - \frac{1}{2s} \int_{\mathbf{x}_1,\mathbf{x}_2} p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)} \hat{s}^{(k)}) (\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a})^H(\mathbf{x}_2 - \mathbf{S}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_1^H\mathbf{a}) d\mathbf{x}_1 d\mathbf{x}_2, \tag{J.72}$$

with

$$C = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{V}|, \tag{J.73}$$

and where

$$\mathbf{S} \triangleq \left[\begin{array}{c} \mathbf{0}^H \\ \mathbf{I} \quad \mathbf{0} \end{array}\right] \qquad \mathbf{c} \triangleq (1, 0, \ldots, 0)^H \tag{J.74}$$

Again, the estimation of $\mathbf{a}$ and s is decoupled in certain situations. We may then send $h(\mathbf{a})$ and $h(s)$ separately:

$$\boxed{\begin{array}{l} e^{h(\mathbf{a})} \propto \mathcal{N}^{-1}\left(\mathbf{a} \,\middle|\, \mathbf{m}_a, \mathbf{W}_a\right), \\ e^{h(s)} \propto \mathrm{Ig}\left(s \,\middle|\, -\frac{1}{2}, \frac{\mathrm{E}[(\mathbf{X}_2 - \hat{\mathbf{A}}^{(k+1)}\mathbf{X}_1)^H(\mathbf{X}_2 - \hat{\mathbf{A}}^{(k+1)}\mathbf{X}_1)]}{2}\right), \end{array}}$$

where

$$\mathbf{W}_a = \frac{1}{\hat{s}^{(k)}} \mathrm{E}\left[\mathbf{X}_1 \mathbf{X}_1^H\right] \tag{J.75}$$

$$\mathbf{m}_a = \frac{1}{\hat{s}^{(k)}} \mathbf{W}_a^{-1} \mathrm{E}\left[\mathbf{X}_1 \left[\mathbf{X}_2\right]_1\right], \tag{J.76}$$

with $[\mathbf{X}_i]_j$ is the j-th component of the (random) vector $\mathbf{X}_i$.

If the estimation of $\mathbf{a}$ and s is coupled, one may still send the messages separately. This corresponds again to approximating the M-step by ICM (cf. Section 4.9.5). If one wishes to perform the exact M-step, one needs to send the (exact) message $e^{h(a,s)}$ defined as:

$$\boxed{e^{h(\mathbf{a},s)} \propto \mathcal{N}^{-1}\Big(\mathbf{a} \,\Big|\, \mathbf{m}_a(s), \mathbf{W}_a(s)\Big),}$$

where

$$\mathbf{W}_a(s) = \frac{1}{s}\,\mathrm{E}\left[\mathbf{X}_1\mathbf{X}_1^H\right] \tag{J.77}$$

$$\mathbf{m}_a(s) = \frac{1}{s}\,\mathbf{W}_a^{-1}\mathrm{E}\left[\mathbf{X}_1\left[\mathbf{X}_2\right]_1\right], \tag{J.78}$$

with $[\mathbf{X}_i]_j$ is the j-th component of the (random) vector $\mathbf{X}_i$.

J.5 Finite State Machine

This node is not used in this thesis. Nonetheless, its derivation is shown here, because it appears in one of the prime applications of the EM algorithm. The function of a node implementing the state transition of

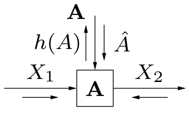


Figure J.6: Factor graph node of the state transition node for the finite state machine.

a finite state machine is (cf. Fig. J.6)

$$f(x_1, x_2, A) = a_{x_1x_2} = \sum_{i,j} a_{ij}\delta[x_1 - i]\delta[x_2 - j], \tag{J.79}$$

where $\delta[.]$ denotes the Kronecker delta. The h-message is

$$h(\mathbf{A}) = \sum_{x_1,x_2} p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log f(x_1, x_2, \mathbf{A}) \tag{J.80}$$

$$= \sum_{x_1,x_2} p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log a_{x_1,x_2}. \tag{J.81}$$

This message can be represented as matrix with individual elements $p(x_1, x_2 \mid \hat{\mathbf{A}}^{(k)}) \log a_{x_1,x_2}$.

To find the estimate $\hat{\mathbf{A}}$, one has to compute the derivations

$$\frac{\partial h(\mathbf{A})}{\partial a_{ij}} + \lambda \frac{\partial g(\mathbf{A})}{\partial a_{ij}} = \frac{p(i,j \mid \hat{\mathbf{A}}^{(k)})}{a_{ij}} - \lambda \overset{!}{=} 0, \tag{J.82}$$

with the constraint

$$g(\mathbf{A}) = 1 - \sum_{\ell} a_{i\ell} = 0. \tag{J.83}$$

Therefore

$$\hat{a}_{ij} = \frac{p(i,j \mid \hat{\mathbf{A}}^{(k)})}{\lambda}. \tag{J.84}$$

To find the value of the Lagrange multiplier λ plug (J.84) into the constraint (J.83)

$$\sum_{\ell} \hat{a}_{ij} = \frac{1}{\lambda} \sum_{\ell} p(i,\ell \mid \hat{\mathbf{A}}^{(k)}) = 1 \tag{J.85}$$

$$\lambda = \sum_{\ell} p(i,\ell \mid \hat{\mathbf{A}}^{(k)}). \tag{J.86}$$

The new estimates of the transition probabilities a_{ij} thus becomes

$$\boxed{\hat{a}_{ij} = \frac{p(i,j \mid \hat{\mathbf{A}}^{(k)})}{\sum_{\ell} p(i,\ell \mid \hat{\mathbf{A}}^{(k)})}} \tag{J.87}$$

J.6 Computing the expectations of Table 4.2

In this section it is explained how the expectations in the message passing EM update rules of Table 4.2 are computed from the sum-product messages when the incoming messages are Gaussian.

J.6.1 Scalar case

The expectation $\mathrm{E}[X_1 X_2 | \hat{a}^{(k)}]$ is derived from the density $p(x_1, x_2 | \hat{a}^{(k)})$ which is given by

$$p(x_1, x_2 | \hat{a}^{(k)}) \propto \mu_{X_1 \to f}(x_1) f(x_1, x_2, \hat{a}^{(k)}) \mu_{X_2 \to f}(x_2) \tag{J.88}$$

$$\propto \mathcal{N}(\mathbf{x} \mid \mathbf{m}_{1,2}, \mathbf{W}_{1,2}) \tag{J.89}$$

where $\mathbf{x} \triangleq (x_2, x_1)^H$, the node function is

$$f(x_1, x_2, \hat{a}^{(k)}) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(x_2 - \hat{a}^{(k)} x_1)^2}{2s} \right) \tag{J.90}$$

and the incoming message $\mu_{X_1 \to f}(x_1)$ is Gaussian with mean m_1 and weight w_1 and similarly m_2 and w_2 for $\mu_{X_2 \to f}(x_2)$. (J.88) can be interpreted as propagating three augmented messages through an equality-node. We, therefore, can apply the update rule of Table H.1 extended to three incoming messages. The mean vectors and weight matrices of the augmented messages are

$$\tilde{\mathbf{m}}_1 = (0, m_1)^H \qquad \tilde{\mathbf{W}}_1 = \begin{bmatrix} 0 & 0 \\ 0 & w_1 \end{bmatrix} \tag{J.91}$$

$$\tilde{\mathbf{m}}_2 = (m_2, 0)^H \qquad \tilde{\mathbf{W}}_2 = \begin{bmatrix} w_2 & 0 \\ 0 & 0 \end{bmatrix} \tag{J.92}$$

$$\tilde{\mathbf{m}}_f = \mathbf{0} \qquad \tilde{\mathbf{W}}_f = \frac{1}{s} \begin{bmatrix} 1 & -a \\ -a & a^2 \end{bmatrix} \tag{J.93}$$

The vector and the matrix (J.93) are found by comparing the coefficients of

$$\frac{(x_2 - a x_1)^2}{s} = (\mathbf{x} - \tilde{\mathbf{m}}_f)^H \tilde{\mathbf{W}}_f (\mathbf{x} - \tilde{\mathbf{m}}_f) \tag{J.94}$$

The joint density (J.88) is a Gaussian with mean vector and weight matrix

$$\mathbf{m}_{1,2} = (\mathbf{W}_1 + \mathbf{W}_f + \mathbf{W}_2)^{-1} (\mathbf{W}_1 \mathbf{m}_1 + \mathbf{W}_f \mathbf{m}_f + \mathbf{W}_2 \mathbf{m}_2) \tag{J.95}$$

$$= \mathbf{W}_{1,2}^{-1} \begin{pmatrix} w_2 m_2 \\ w_1 m_1 \end{pmatrix} \tag{J.96}$$

$$\mathbf{W}_{1,2} = \mathbf{W}_1 + \mathbf{W}_f + \mathbf{W}_2. \tag{J.97}$$

The correlation matrix finally is

$$\begin{bmatrix} \mathrm{E}[X_2^2] & \mathrm{E}[X_1 X_2] \\ \mathrm{E}[X_2 X_1] & \mathrm{E}[X_1^2] \end{bmatrix} = \mathbf{W}_{1,2}^{-1} + \mathbf{m}_{1,2} \mathbf{m}_{1,2}^H. \tag{J.98}$$

J.6.2 Vector case

Here the joint density of X_1 and X_2 given $\hat{\mathbf{a}}^{(k)}$ is

$$\begin{aligned} p(\mathbf{x}_1, \mathbf{x}_2 | \hat{\mathbf{a}}^{(k)}) &\propto \mu_{\mathbf{X}_1 \to f}(\mathbf{x}_1) f(\mathbf{x}_1, \mathbf{x}_2, \hat{\mathbf{a}}^{(k)}) \mu_{\mathbf{X}_2 \to f}(\mathbf{x}_2) && \text{(J.99)} \\ &\propto \mathcal{N}(\mathbf{x} \mid \mathbf{m}_{1,2}, \mathbf{W}_{1,2}) && \text{(J.100)} \end{aligned}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}$ are the following vectors:

$$\begin{aligned} \mathbf{x}_1 &= (x_{n-1}, x_{n-2}, \ldots, x_{n-M})^H && \text{(J.101)} \\ \mathbf{x}_2 &= (x_n, x_{n-1}, \ldots, x_{n-M+1})^H && \text{(J.102)} \\ \mathbf{x} &= (x_n, x_{n-1}, \ldots, x_{n-M+1}, x_{n-M})^H && \text{(J.103)} \end{aligned}$$

The augmented mean vectors and weight matrices are

$$\begin{aligned} \tilde{\mathbf{m}}_1 &= (0, \mathbf{m}_1)^H \qquad \tilde{\mathbf{W}}_1 = \begin{bmatrix} 0 & \mathbf{0}^H \\ \mathbf{0} & \mathbf{W}_1 \end{bmatrix} && \text{(J.104)} \\ \tilde{\mathbf{m}}_2 &= (\mathbf{m}_2, 0)^H \qquad \tilde{\mathbf{W}}_2 = \begin{bmatrix} \mathbf{W}_2 & \mathbf{0}^H \\ \mathbf{0} & 0 \end{bmatrix} && \text{(J.105)} \\ \tilde{\mathbf{m}}_f &= \mathbf{0} && \text{(J.106)} \end{aligned}$$

$$\tilde{\mathbf{W}}_f = \frac{1}{s} \left[\begin{array}{c|cccc} 1 & -a_1 & \cdots & & -a_n \\ \hline -a_1 & a_1^2 & a_1 a_2 & \cdots & \\ \vdots & a_1 a_2 & \ddots & & \vdots \\ & \vdots & & a_{n-1}^2 & a_n a_{n-1} \\ -a_n & & \cdots & a_n a_{n-1} & a_n^2 \end{array} \right] \qquad \text{(J.107)}$$

$$= \frac{1}{s} \left[\begin{array}{c|c} 1 & -\mathbf{a}^H \\ \hline -\mathbf{a} & \mathbf{a}\mathbf{a}^H \end{array} \right] \qquad \text{(J.108)}$$

The vector (J.106) and the matrix (J.108) are found by comparing the coefficients of

$$\frac{(x_n - \sum_{k=1}^{n} a_k x_{n-k})^2}{s} = (\tilde{\mathbf{x}} - \tilde{\mathbf{m}}_f)^H \tilde{\mathbf{W}}_f (\tilde{\mathbf{x}} - \tilde{\mathbf{m}}_f) \qquad \text{(J.109)}$$

The joint density (J.99) is Gaussian with mean vector and weight matrix

$$\begin{aligned} \mathbf{m}_{1,2} &= (\tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_f + \tilde{\mathbf{W}}_2)^{-1} (\tilde{\mathbf{W}}_1 \tilde{\mathbf{m}}_1 + \tilde{\mathbf{W}}_f \tilde{\mathbf{m}}_f + \tilde{\mathbf{W}}_2 \tilde{\mathbf{m}}_2) && \text{(J.110)} \\ &= \mathbf{W}_{1,2}^{-1} \left[\begin{pmatrix} 0 \\ \mathbf{W}_1 \mathbf{m}_1 \end{pmatrix} + \begin{pmatrix} \mathbf{W}_2 \mathbf{m}_2 \\ 0 \end{pmatrix} \right] && \text{(J.111)} \\ \mathbf{W}_{1,2} &= \tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_f + \tilde{\mathbf{W}}_2. && \text{(J.112)} \end{aligned}$$

The correlation matrix finally is

$$\begin{bmatrix} \mathrm{E}[X_2^2] & \mathrm{E}[\mathbf{X}_1^H X_2] \\ \mathrm{E}[X_2 \mathbf{X}_1] & \mathrm{E}[\mathbf{X}_1 \mathbf{X}_1^H] \end{bmatrix} = \mathbf{W}_{1,2}^{-1} + \mathbf{m}_{1,2} \mathbf{m}_{1,2}^H. \tag{J.113}$$

Appendix K

Mathematical Background of Chapter 5

Lemma K.1. Let $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \succeq 0$, where $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, $\mathbf{A}_{21}$, and $\mathbf{A}_{22}$ are $n \times n$, $n \times m$, $m \times n$, and $m \times m$ submatrices respectively. Let $\mathbf{A}_{22}$ be nonsingular.
Then $(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}) \succeq 0$. $\square$

Proof: $\mathbf{A} \succeq 0$, which by definition means that $v^T\mathbf{A}v \geq 0, \forall v \in \mathbb{R}^{n+m}$.
Let $v = [v_1 v_2]^T$, where $v_1 \in \mathbb{R}^n$ and $v_2 \in \mathbb{R}^m$.
Then $v^T\mathbf{A}v = v_1^T\mathbf{A}_{11}v_1 + v_1^T\mathbf{A}_{12}v_2 + v_2^T\mathbf{A}_{21}v_1 + v_2^T\mathbf{A}_{22}v_2 \geq 0$.
Define now $v_2^T \triangleq -v_1^T\mathbf{A}_{12}\mathbf{A}_{22}^{-1}$, then

$$\begin{aligned} v^T\mathbf{A}v &= v_1^T\mathbf{A}_{11}v_1 + v_1^T\mathbf{A}_{12}v_2 + v_2^T\mathbf{A}_{21}v_1 + v_2^T\mathbf{A}_{22}v_2 && \text{(K.1)} \\ &= v_1^T\mathbf{A}_{11}v_1 + v_1^T\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{22}v_2 + v_2^T\mathbf{A}_{21}v_1 + v_2^T\mathbf{A}_{22}v_2 && \text{(K.2)} \\ &= v_1^T\mathbf{A}_{11}v_1 - v_2^T\mathbf{A}_{22}v_2 + v_2^T\mathbf{A}_{21}v_1 + v_2^T\mathbf{A}_{22}v_2 && \text{(K.3)} \\ &= v_1^T(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})v_1 \geq 0, \forall v_1 \in R^n. && \text{(K.4)} \end{aligned}$$

In (K.2), we have used the fact that $\mathbf{A}_{22}$ is nonsingular.
The equalities (K.3) and (K.4) follow from the definition of v_2.
As a consequence of (K.4), $(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}) \succeq 0$. $\square$

Lemma K.2. (Matrix inversion Lemma)
Let $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$, where $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are nonsingular $n \times n$ and $m \times m$ submatrices respectively, such that $(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})$ and $(\mathbf{A}_{11} - \mathbf{A}_{22}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})$ are also nonsingular.
Then $\mathbf{A}$ is also nonsingular with

$$\begin{aligned} \left[\mathbf{A}^{-1}\right]_{11} &= \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} && \text{(K.5)} \\ &= (\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1} && \text{(K.6)} \end{aligned}$$

$$\begin{aligned} \left[\mathbf{A}^{-1}\right]_{12} &= -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1} && \text{(K.7)} \\ &= -(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} && \text{(K.8)} \end{aligned}$$

$$\begin{aligned} \left[\mathbf{A}^{-1}\right]_{21} &= -(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} && \text{(K.9)} \\ &= -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1} && \text{(K.10)} \end{aligned}$$

$$\begin{aligned} \left[\mathbf{A}^{-1}\right]_{22} &= \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{21}(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} && \text{(K.11)} \\ &= (\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}. && \text{(K.12)} \end{aligned}$$

□

Proof: We refer to [33, pp. 8–9]. □

Lemma K.3. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$.

$$\mathbf{A} \succ 0 \Leftrightarrow \mathbf{B}^T\mathbf{A}\mathbf{B} \succ 0. \tag{K.13}$$

□

Proof:

$\boxed{\Rightarrow}$

$\mathbf{A} \succ 0$ means by definition that $v^T\mathbf{A}v > 0$, for all $v \in \mathbb{R}^n$. In particular, define $v \triangleq \mathbf{B}u$ with $u \in \mathbb{R}^n$. As a consequence, $u^T\mathbf{B}^T\mathbf{A}\mathbf{B}u > 0$, for all $u \in \mathbb{R}^n$. Hence, $\mathbf{B}^T\mathbf{A}\mathbf{B} \succ 0$.

$\boxed{\Leftarrow}$

$\mathbf{B}^T\mathbf{A}\mathbf{B} \succ 0$ means by definition that $v^T\mathbf{B}^T\mathbf{A}\mathbf{B}v > 0$, for all $v \in \mathbb{R}^n$. Defining $u \triangleq \mathbf{B}u$, we have $u^T\mathbf{A}u > 0$, for all $u \in \mathbb{R}^n$. Hence, $\mathbf{A} \succ 0$.

□

Lemma K.4. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$, and suppose that $\mathbf{A} \succ 0$, $\mathbf{B} \succ 0$, $\mathbf{A} \succeq \mathbf{B}$ and $\mathbf{B}^T = \mathbf{B}$. Then:

$$\mathbf{B}^{-1} \succeq \mathbf{A}^{-1}. \tag{K.14}$$

□

Proof: $\mathbf{A} \succeq \mathbf{B}$ means that $\mathbf{D} \triangleq \mathbf{A} - \mathbf{B} \succeq 0$. We show now that $\mathbf{F} \triangleq \mathbf{B}^{-1} - \mathbf{A}^{-1} \triangleq \mathbf{B}^{-1} - (\mathbf{B}+\mathbf{D})^{-1}$ is positive semi-definite. First, note that $\mathbf{F}$ is well-defined, since positive definite matrices are non-singular. Note also that:

$$\begin{aligned}
&(\mathbf{B}+\mathbf{D})(\mathbf{B}^{-1} - \mathbf{A}^{-1})(\mathbf{B}+\mathbf{D})^T \\
&\qquad = (\mathbf{B}+\mathbf{D})\left(\mathbf{B}^{-1} - (\mathbf{B}+\mathbf{D})^{-1}\right)(\mathbf{B}+\mathbf{D})^T && \text{(K.15)}\\
&\qquad = (\mathbf{B}+\mathbf{D})\mathbf{B}^{-1}(\mathbf{B}+\mathbf{D})^T - (\mathbf{B}+\mathbf{D})^T && \text{(K.16)}\\
&\qquad = (\mathbf{B}+\mathbf{D})^T + \mathbf{D}\mathbf{B}^{-1}(\mathbf{B}+\mathbf{D})^T - (\mathbf{B}+\mathbf{D})^T && \text{(K.17)}\\
&\qquad = \mathbf{D}\mathbf{B}^{-1}(\mathbf{B}+\mathbf{D})^T && \text{(K.18)}\\
&\qquad = \mathbf{D}\mathbf{B}^{-1}\mathbf{B}^T + \mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T && \text{(K.19)}\\
&\qquad = \mathbf{D} + \mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T. && \text{(K.20)}
\end{aligned}$$

In (K.20), we have used the fact that $\mathbf{B}^T = \mathbf{B}$. Since the RHS of (K.20) is positive semi-definite (cf. Lemma K.3), the same holds for the LHS of (K.15). From Lemma K.3, we conclude that $\mathbf{F}$ is positive semi-definite, hence, $\mathbf{B}^{-1} \succeq \mathbf{A}^{-1}$. □

Lemma K.5. Let $\mathbf{A}$ be a nonsingular $n \times n$ matrix. Then

$$\left([\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}]^{-1}\right)_{kk} = \left[\mathbf{A}^{-1}\right]_{kk}, \quad (k \neq i, k \neq j) \tag{K.21}$$

and

$$\left([\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}]^{-1}\right)_{ii} = \left[\mathbf{A}^{-1}\right]_{jj}, \tag{K.22}$$

where $\mathbf{f}^{ij}$ is the permutation matrix obtained by permuting the ith and jth row in the $n \times n$ unity matrix. □

Proof: Note that the matrix $\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}$ is obtained from $\mathbf{A}$ by permuting the ith and jth row and column. It is well-known that

$$\mathbf{A}^{-1} = \frac{\text{adjoint}\mathbf{A}}{\det\mathbf{A}}, \tag{K.23}$$

where

$$(\text{adjoint}\mathbf{A})_{ij} = (-1)^{i+j}\det\mathbf{A}^{(ji)}, \tag{K.24}$$

and $\mathbf{A}^{(ji)}$ is obtained from $\mathbf{A}$ by erasing the ith row and jth column. It holds that

$$\det(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}) = \det{}^2(\mathbf{f}^{ij})\det(\mathbf{A}) = (1)^2\det(\mathbf{A}) = \det(\mathbf{A}), \tag{K.25}$$

$$\begin{aligned}
[\text{adjoint}(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})]_{kk} &= (-1)^{2k}\det[(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})^{(kk)}] && \text{(K.26)}\\
&= \det\left[((\mathbf{f}^{ij})^{(kk)}\mathbf{A}^{(kk)}(\mathbf{f}^{ij})^{(kk)})\right] && \text{(K.27)}\\
&= \det\left[\mathbf{A}^{(kk)}\right] && \text{(K.28)}\\
&= [\text{adjoint}\mathbf{A}]_{kk}, \quad (k \neq i, k \neq j) && \text{(K.29)}
\end{aligned}$$

and

$$\begin{aligned}
[\text{adjoint}(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})]_{ii} &= (-1)^{2j}\det[(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})^{(ii)}] && \text{(K.30)}\\
&= \det\left[\mathbf{A}^{(jj)}\right] && \text{(K.31)}\\
&= [\text{adjoint}\mathbf{A}]_{jj}. && \text{(K.32)}
\end{aligned}$$

As a consequence

$$\begin{aligned}
\left([\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}]^{-1}\right)_{kk} &= \frac{\text{adjoint}(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})_{kk}}{\det(\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij})} && \text{(K.33)}\\
&= \frac{(\text{adjoint}\mathbf{A})_{kk}}{\det\mathbf{A}} && \text{(K.34)}\\
&= [\mathbf{A}^{-1}]_{kk}. \quad (k \neq i, k \neq j) && \text{(K.35)}
\end{aligned}$$

and similarly

$$\left([\mathbf{f}^{ij}\mathbf{A}\mathbf{f}^{ij}]^{-1}\right)_{ii} = [\mathbf{A}^{-1}]_{jj}. \tag{K.36}$$

□

Lemma K.6. If

a) $\nabla_{\theta_j} p(y|\theta)$ and $\nabla_{\theta_i}\nabla_{\theta_j}^T p(y|\theta)$ exist $\forall\theta$ and y,

b) $\mathrm{E}_{Y|\Theta}\left[-\nabla_{\theta_i}\nabla_{\theta_j}^T \log p(Y|\theta)\right]$ and $\mathrm{E}_{Y|\Theta}\left[\nabla_{\theta_i}\log p(Y|\theta)\nabla_{\theta_j}^T \log p(Y|\theta)\right]$ exist,

then $\mathrm{E}_{Y|\Theta}\left[-\nabla_{\theta_i}\nabla^T_{\theta_j}\log p(Y|\theta)\right] = \mathrm{E}_{Y|\Theta}\left[\nabla_{\theta_i}\log p(Y|\theta)\nabla^T_{\theta_j}\log p(Y|\theta)\right]$.
□

Proof: Differentiate both sides of the equation $\int_y p(y|\theta)dy = 1$ w.r.t. θ_j. From Assumption 1, it follows

$$\int_y \nabla^T_{\theta_j} p(y|\theta)dy = \int_y p(y|\theta)\nabla^T_{\theta_j}\log p(y|\theta)dy = 0.$$

and

$$\int_y p(y|\theta)\nabla_{\theta_i}\nabla^T_{\theta_j}\log p(y|\theta)dy + \int_y \nabla_{\theta_i} p(y|\theta)\nabla^T_{\theta_j}\log p(y|\theta)dy = 0.$$

Therefore,

$$\mathrm{E}_{Y|\Theta}[\nabla_{\theta_i}\nabla^T_{\theta_j}\log p(Y|\theta)] = -\mathrm{E}_{Y|\Theta}[\nabla_{\theta_i}\log p(Y|\theta)\nabla^T_{\theta_j}\log p(Y|\theta)],$$

where the expectations in both sides are well-defined (Assumption 2). □

Lemma K.7. If

a) $\nabla_{x_j} p(x,y)$ and $\nabla_{x_i}\nabla^T_{x_j} p(x,y)$ exist $\forall x$ and y,

b) $\mathrm{E}_{XY}\Big[\nabla_{x_i}\log p(X,Y)\nabla^T_{x_j}\log p(X,Y)\Big]$
and $\mathrm{E}_{XY}\Big[-\nabla_{x_i}\nabla^T_{x_j}\log p(X,Y)\Big]$ exist,

c) $\int_{x,y}\nabla_{x_i}\nabla^T_{x_j} p(x,y)dxdy = 0$,

then $\mathrm{E}_{XY}\Big[-\nabla_{x_i}\nabla^T_{x_j}\log p(X,Y)\Big] = \mathrm{E}_{XY}\Big[\nabla_{x_i}\log p(X,Y)\nabla^T_{x_j}\log p(X,Y)\Big]$.
□

Proof: From Assumption 1, it follows

$$\nabla^T_{x_j} p(x,y) = p(x,y)\nabla^T_{x_j}\log p(x,y), \tag{K.37}$$

and

$$\nabla_{x_i}\nabla^T_{x_j} p(x,y) = \nabla_{x_i}\big(p(x,y)\nabla^T_{x_j}\log p(x,y)\big) \tag{K.38}$$
$$= p(x,y)\nabla_{x_i}\log p(x,y)\nabla^T_{x_j}\log p(x,y)$$
$$+p(x,y)\nabla_{x_i}\nabla^T_{x_j}\log p(x,y). \tag{K.39}$$

Integrating both sides over X and Y, one obtains

$$\int_x \int_y p(x,y)\nabla_{x_i} \log p(x,y) \nabla_{x_j}^T \log p(x,y) dx dy \tag{K.40}$$

$$+\int_x \int_y p(x,y)\nabla_{x_i} \nabla_{x_j}^T \log p(x,y) dx dy \tag{K.41}$$

$$= 0, \tag{K.42}$$

where (K.42) follows from Assumption 3.

As a consequence

$$\mathrm{E}_{XY}[\nabla_{x_i}\nabla_{x_j}^T \log p(X,Y)]$$
$$= -\mathrm{E}_{XY}[\nabla_{x_i} \log p(X,Y) \nabla_{x_j}^T \log p(X,Y)], \tag{K.43}$$

where the expectations in both sides are well-defined (Assumption 2). □

Lemma K.8. If

a) $\nabla_{x_j} p(y|x)$ and $\nabla_{x_i}\nabla_{x_j}^T p(y|x)$ exist $\forall x$ and y,

b) $\mathrm{E}_{XY}\left[\nabla_{x_i} \log p(Y|X) \nabla_{x_j}^T \log p(Y|X)\right]$
and $\mathrm{E}_{XY}\left[-\nabla_{x_i}\nabla_{x_j}^T \log p(Y|X)\right]$ exist,

then $\mathrm{E}_{XY}\left[-\nabla_{x_i}\nabla_{x_j}^T \log p(Y|X)\right] = \mathrm{E}_{XY}\left[\nabla_{x_i} \log p(y|x) \nabla_{x_j}^T \log p(Y|X)\right]$.
□

Proof: Differentiate both sides of the equation $\int_y p(y|x)dy = 1$ w.r.t. x_j. From Assumption 1, it follows

$$\int_y \nabla_{x_j}^T p(y|x) dy = \int_y p(y|x) \nabla_{x_j}^T \log p(y|x) dy = 0.$$

and

$$\int_y p(y|x)\nabla_{x_i}\nabla_{x_j}^T \log p(y|x) dy + \int_y \nabla_{x_i} p(y|x) \nabla_{x_j}^T \log p(y|x) dy = 0.$$

Multiplying both sides with $p(x)$ and integrating over x, we obtain:

$$\mathrm{E}_{XY}[\nabla_{x_i}\nabla_{x_j}^T \log p(Y|X)] = -\mathrm{E}_{XY}[\nabla_{x_i} \log p(Y|X) \nabla_{x_j}^T \log p(Y|X)],$$

where the expectations in both sides are well-defined (Assumption 2). □

Lemma K.9. If

a) $\nabla_{x_j} p(x)$ and $\nabla_{x_i} \nabla_{x_j}^T p(x)$ exist $\forall x$,

b) $\mathrm{E}_X \left[-\nabla_{x_i} \nabla_{x_j}^T \log p(X) \right]$ and $\mathrm{E}_X \left[\nabla_{x_i} \log p(X) \nabla_{x_j}^T \log p(X) \right]$ exist,

c) $\int_x \nabla_{x_i} \nabla_{x_j}^T p(x) dx = 0$,

then $\mathrm{E}_X \left[-\nabla_{x_i} \nabla_{x_j}^T \log p(X) \right] = \mathrm{E}_X \left[\nabla_{x_i} \log p(X) \nabla_{x_j}^T \log p(X) \right]$. □

Proof: As a consequence of Assumption 1, $\nabla_{x_j}^T p(x) = p(x) \nabla_{x_j}^T \log p(x)$ and

$$\begin{aligned} \nabla_{x_i} \nabla_{x_j}^T p(x) &= \nabla_{x_i} \left(p(x) \nabla_{x_j}^T \log p(x) \right) \\ &= p(x) \nabla_{x_i} \log p(x) \nabla_{x_j}^T \log p(x) + p(x) \nabla_{x_i} \nabla_{x_j}^T \log p(x). \end{aligned}$$

If one integrates both sides over x, as a consequence of Assumption 3, one obtains:

$$\int_x p(x) \nabla_{x_i} \log p(x) \nabla_{x_j}^T \log p(x) dx + \int_x \int_y p(x) \nabla_{x_i} \nabla_{x_j}^T \log p(x) dx = 0.$$

Therefore,

$$\mathrm{E}_X [\nabla_{x_i} \nabla_{x_j}^T \log p(X)] = -\mathrm{E}_X [\nabla_{x_i} \log p(X) \nabla_{x_j}^T \log p(X)],$$

where the expectations in both sides are well-defined (Assumption 2). □

Lemma K.10. Let $\mathbf{A}$ be a random square matrix, almost surely positive definite. Then,

$$(\mathrm{E}[\mathbf{A}])^{-1} \preceq \mathrm{E}\left[\mathbf{A}^{-1}\right]. \tag{K.44}$$

□

For a proof, we refer to [24].

Proof of Theorem 5.1

The estimator $\hat{\theta}(y)$ is supposed to be unbiased (Assumption f), i.e.,

$$B(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy = 0. \tag{K.45}$$

Differentiating both sides with respect to Θ, we have

$$\begin{aligned}
\nabla_{\theta_i} \int_y [\hat{\theta}_j(y) - \theta_j] p(y|\theta) dy &= \int_y \nabla_{\theta_i} \big([\hat{\theta}_j(y) - \theta_j] p(y|\theta) \big) dy && \text{(K.46)} \\
&= -\delta_{ij} \int_y p(y|\theta) dy \\
&\quad + \int_y [\hat{\theta}_j(y) - \theta_j] \nabla_{\theta_i} p(y|\theta) dy && \text{(K.47)} \\
&= -\delta_{ij} + \int_y [\hat{\theta}_j(y) - \theta_j] \nabla_{\theta_i} p(y|\theta) dy && \text{(K.48)} \\
&= 0. && \text{(K.49)}
\end{aligned}$$

The equality (K.46) follows from Assumption c and e. Note that if the integration limits in $B(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy$ depended on θ (cf. Assumption c), additional terms would appear in the RHS of (K.46) (cf. Theorem I.2). The equality (K.48) follows from the fact that $p(y|\theta)$ is a probability function in Y.

As a consequence of Assumption d:

$$\nabla_\theta p(y|\theta) = \nabla_\theta \log p(y|\theta) / p(y|\theta). \tag{K.50}$$

Substituting (K.51) in (K.48), one obtains:

$$\delta_{ij} = \int_y [\hat{\theta}_j(y) - \theta_j] p(y|\theta) \nabla_{\theta_i} \log p(y|\theta) dy. \tag{K.51}$$

Now, we define the vector v as

$$v \triangleq [\hat{\theta}_1(y) - \theta_1, \ldots, \hat{\theta}_n(y) - \theta_n, \nabla_{\theta_1} \log p(y|\theta), \ldots, \nabla_{\theta_n} \log p(y|\theta)]^T, \tag{K.52}$$

and the matrix $\mathbf{C}_v$ as

$$\mathbf{C}_v \triangleq \mathrm{E}_{Y|\Theta}[vv^T] = \begin{bmatrix} \mathbf{E}(\theta) & \mathbf{I} \\ \mathbf{I} & \mathbf{F}(\theta) \end{bmatrix}, \tag{K.53}$$

where $\mathbf{I}$ is a unity matrix. The RHS of (K.53) follows from (5.1), (5.2), and (K.51). The matrix $\mathbf{C}_v$ is well-defined as a consequence of Assumption 1 and 2. Note that $\mathbf{C}_v \succeq 0$, since $u^T\mathbf{C}_v u = u^T \mathrm{E}_v[vv^T]u = \mathrm{E}_v[u^T vv^T u] = \mathrm{E}_v[|v^T u|^2] \geq 0, \forall u$. Since $\mathbf{C}_v \succeq 0$ and $\mathbf{F}(\theta)$ is non-singular (cf. Assumption 2), we can apply Lemma K.1; hence, $\mathbf{E}(\theta) \succeq \mathbf{F}^{-1}(\theta)$. □

Proof of Theorem 5.3

First, note that:

$$\begin{aligned}
\nabla_{x_j}[p(x)B_i(x)] &= \nabla_{x_j}\left[p(x)\int_y [\hat{x}_i(y) - x_i]p(y|x)dy\right] && \text{(K.54)}\\
&= \nabla_{x_j}\left[\int_y [\hat{x}_i(y) - x_i]p(x,y)dy\right] && \text{(K.55)}\\
&= -\delta_{ij}\int_y p(x,y)dy \\
&\quad + \int_y [\hat{x}_i(y) - x_i]\nabla_{x_j}p(x,y)dy. && \text{(K.56)}
\end{aligned}$$

In the equality (K.54), we used the definition of $B(x)$. The equality (K.56) follows from Assumption c and e. Note that if the integration limits in $B(x) \stackrel{\triangle}{=} \int_y [\hat{x}(y) - x]p(y|x)dy$ depended on x (cf. Assumption c), additional terms would appear in the RHS of (K.56) (cf. Theorem I.2).
Now we integrate both sides of (K.56) with respect to x

$$\begin{aligned}
&\int_x \nabla_{x_j}[B_i(x)p(x)]dx && \text{(K.57)}\\
&= -\delta_{ij}\int_{x,y} p(x,y)dxdy + \int_{x,y}[\hat{x}_i(y) - x_i]\nabla_{x_j}p(x,y)dxdy && \text{(K.58)}\\
&= -\delta_{ij} + \int_{x,y}[\hat{x}_i(y) - x_i]\nabla_{x_j}p(x,y)dxdy && \text{(K.59)}\\
&= -\delta_{ij} + \int_{x,y}[\hat{x}_i(y) - x_i]p(x,y)\nabla_{x_j}\log p(x,y)dxdy && \text{(K.60)}\\
&= 0. && \text{(K.61)}
\end{aligned}$$

The equality (K.59) follows from the fact that $p(x, y)$ is a probability function; the equality (K.61) follows from Assumption d.
We define the vector v as

$$v \triangleq [\hat{x}_1(y)-x_1, \ldots, \hat{x}_n(y)-x_n, \nabla_{x_1} \log p(x, y), \ldots, \nabla_{x_n} \log p(x, y)]^T, \tag{K.62}$$

and the matrix $\mathbf{C}_v$ as

$$\mathbf{C}_v \triangleq \mathrm{E}_{XY}[vv^T] = \begin{bmatrix} \mathbf{E} & \mathbf{I} \\ \mathbf{I} & \mathbf{J} \end{bmatrix}, \tag{K.63}$$

where $\mathbf{I}$ is a unity matrix; the matrix $\mathbf{C}_v$ is well-defined as a consequence of Assumption 1 and 2. Since $\mathbf{C}_v \succeq 0$ and $\mathbf{J}$ is non-singular (cf. Assumption 2), we can apply Lemma K.1; hence, $\mathbf{E} \succeq \mathbf{J}^{-1}$. □

Proof of Theorem 5.4

First, note that:

$$\nabla_{x_j} [(\hat{x}_i(y) - x_i)p(x|y)] = -\delta_{ij} p(x|y) + [\hat{x}_i(y) - x_i] \nabla_{x_j} p(x|y). \tag{K.64}$$

As a consequence of Assumption c, the derivatives in (K.64) are well-defined. We now integrate both sides of (K.64) with respect to x

$$\int_x \nabla_{x_j} [(\hat{x}_i(y) - x_i)p(x|y)]\, dx \tag{K.65}$$

$$= -\delta_{ij} \int_x p(x|y) dx + \int_x [\hat{x}_i(y) - x_i] \nabla_{x_j} p(x|y) dx \tag{K.66}$$

$$= -\delta_{ij} + \int_x [\hat{x}_i(y) - x_i] \nabla_{x_j} p(x|y) dx \tag{K.67}$$

$$= -\delta_{ij} + \int_x [\hat{x}_i(y) - x_i] p(x|y) \nabla_{x_j} \log p(x|y) dx \tag{K.68}$$

$$= 0. \tag{K.69}$$

The equality (K.67) follows from the fact that $p(x|y)$ is a probability function; the equality (K.69) follows from Assumption d.
We define the vector v as

$$v \triangleq [\hat{x}_1(y)-x_1, \ldots, \hat{x}_n(y)-x_n, \nabla_{x_1} \log p(x|y), \ldots, \nabla_{x_n} \log p(x|y)]^T, \tag{K.70}$$

and the matrix $\mathbf{C}_v$ as

$$\mathbf{C}_v \triangleq \mathrm{E}_{X|Y}[vv^T] = \begin{bmatrix} \mathbf{E}(y) & \mathbf{I} \\ \mathbf{I} & \mathbf{J}(y) \end{bmatrix}, \tag{K.71}$$

where $\mathbf{I}$ is a unity matrix; the matrix $\mathbf{C}_v$ is well-defined as a consequence of Assumption 1 and 2. Since $\mathbf{C}_v \succeq 0$ and $\mathbf{J}(y)$ is non-singular (cf. Assumption 2), we can apply Lemma K.1; hence, $\mathbf{E}(y) \succeq \mathbf{J}^{-1}(y)$. $\square$

Proof of Lemma 5.1

Note that

$$\mathbf{J} = \mathrm{E}_Y[\mathbf{J}(Y)]. \tag{K.72}$$

The inequality (5.33) follows from (K.72) and Lemma K.10. $\square$

Proof of Theorem 5.5

Note that the inequality (5.5) (with Θ replaced by X) holds for all x. One obtains the inequality (5.34) by first multiplying both sides of (5.5) with $p(x)$, and by then integrating both sides over x. $\square$

Proof of Lemma 5.2

Note that

$$\mathbf{J} = \mathrm{E}_X[\mathbf{F}(X)] + \mathrm{E}_X\left[\nabla_x \log p(X) \nabla_x^T \log p(X)\right], \tag{K.73}$$

and $\mathrm{E}_X\left[\nabla_x \log p(X) \nabla_x^T \log p(X)\right] \succeq 0$. Therefore,

$$\mathbf{J} \succeq \mathrm{E}_X[\mathbf{F}(X)], \tag{K.74}$$

and, as a consequence of Lemma K.4,

$$\mathbf{J}^{-1} \preceq \left(\mathrm{E}_X[\mathbf{F}(X)]\right)^{-1}. \tag{K.75}$$

From Lemma K.10, we have:

$$\left(\mathrm{E}_X[\mathbf{F}(X)]\right)^{-1} \preceq \mathrm{E}_X\left[\mathbf{F}^{-1}(X)\right]. \tag{K.76}$$

The inequality (5.35) follows from (K.75) and (K.76). $\square$

Proof of Theorem 5.6

The following proof is similar to a proof by Reuven et al. [170] for a related bound, i.e., the hybrid Barankin bound; the hybrid CRB is a particular instance of the hybrid Barankin bound, as shown in [170].

First, note that:

$$\begin{aligned}
\nabla_{x_j}\left[p(x)B_i^{(X)}(x)\right] &= \nabla_{x_j}\left[p(x)\int_y [\hat{x}_i(y)-x_i]p(y|x,\theta)dy\right] && \text{(K.77)}\\
&= \nabla_{x_j}\left[\int_y [\hat{x}_i(y)-x_i]p(x,y|\theta)dy\right] && \text{(K.78)}\\
&= -\delta_{ij}\int_y p(x,y|\theta)dy\\
&\qquad + \int_y [\hat{x}_i(y)-x_i]\nabla_{x_j}p(x,y|\theta)dy. && \text{(K.79)}
\end{aligned}$$

In the equality (K.77), we used the definition of $B^{(X)}(x)$. The equality (K.79) follows from Assumption d and e. Note that if the integration limits in $B^{(X)}(x) \triangleq \int_y [\hat{x}(y)-x]p(y|x)dy$ depended on x (cf. Assumption c), additional terms would appear in the RHS of (K.79) (cf. Theorem I.2).
Now we integrate both sides of (K.79) with respect to x

$$\begin{aligned}
&\int_x \nabla_{x_j}[B_i^{(X)}(x)p(x)]dx && \text{(K.80)}\\
&= -\delta_{ij}\int_{x,y} p(x,y|\theta)dxdy + \int_{x,y}[\hat{x}_i(y)-x_i]\nabla_{x_j}p(x,y|\theta)dxdy && \text{(K.81)}\\
&= -\delta_{ij} + \int_{x,y}[\hat{x}_i(y)-x_i]\nabla_{x_j}p(x,y|\theta)dxdy && \text{(K.82)}\\
&= -\delta_{ij} + \int_{x,y}[\hat{x}_i(y)-x_i]p(x,y|\theta)\nabla_{x_j}\log p(x,y|\theta)dxdy && \text{(K.83)}\\
&= 0. && \text{(K.84)}
\end{aligned}$$

The equality (K.82) follows from the fact that $p(x,y)$ is a probability function; the equality (K.84) follows from Assumption f. As a consequence,

$$\int_{x,y} [\hat{x}_i(y) - x_i] p(x,y|\theta) \nabla_{x_j} \log p(x,y|\theta) dx dy = \delta_{ij}. \tag{K.85}$$

The estimator $\hat{\theta}(y)$ is supposed to be unbiased (Assumption g), i.e.,

$$B^{(\Theta)}(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(y|\theta) dy = \int_{x,y} [\hat{\theta}(y) - \theta] p(x,y|\theta) dx dy = 0. \tag{K.86}$$

Differentiating both sides with respect to Θ, we have

$$\begin{aligned}
\nabla_{\theta_i} & \int_{x,y} [\hat{\theta}_j(y) - \theta_j] p(x,y|\theta) dx dy \\
&= \int_{x,y} \nabla_{\theta_i} \big([\hat{\theta}_j(y) - \theta_j] p(x,y|\theta) \big) dx dy & \text{(K.87)} \\
&= -\delta_{ij} + \int_{x,y} [\hat{\theta}_j(y) - \theta_j] \nabla_{\theta_i} p(x,y|\theta) dx dy & \text{(K.88)} \\
&= 0. & \text{(K.89)}
\end{aligned}$$

The equality (K.87) follows from Assumption c and h. Note that if the integration limits in $B^{(\Theta)}(\theta) \triangleq \int_y [\hat{\theta}(y) - \theta] p(x,y|\theta) dx dy$ depended on θ (cf. Assumption c), additional terms would appear in the RHS of (K.87) (cf. Theorem I.2).

As a consequence,

$$\int_{x,y} [\hat{\theta}_j(y) - \theta_j] \nabla_{\theta_i} p(x,y|\theta) dx dy = \delta_{ij}. \tag{K.90}$$

Note that

$$\begin{aligned}
& \int_{x,y} [\hat{\theta}_j(y) - \theta_j] p(x,y|\theta) \nabla_{x_i} \log p(x,y|\theta) dx dy & \text{(K.91)} \\
&= \int_{x,y} [\hat{\theta}_j(y) - \theta_j] \nabla_{x_i} p(x,y|\theta) dx dy & \text{(K.92)} \\
&= \int_y [\hat{\theta}_j(y) - \theta_j] dy \int_x \nabla_{x_i} p(x,y|\theta) dx & \text{(K.93)} \\
&= 0. & \text{(K.94)}
\end{aligned}$$

The equality (K.94) follows from Assumption f.

From Assumption i, it follows:

$$\begin{aligned}
&\nabla_{\theta_j}\left[\int_{x,y}[\hat{x}_i(y)-x_i]p(x,y|\theta)dxdy\right] && \text{(K.95)}\\
&=\int_{x,y}[\hat{x}_i(y)-x_i]\nabla_{\theta_j}p(x,y|\theta)dxdy && \text{(K.96)}\\
&=\int_{x,y}[\hat{x}_i(y)-x_i]p(x,y|\theta)\nabla_{\theta_j}\log p(x,y|\theta)dxdy && \text{(K.97)}\\
&=0. && \text{(K.98)}
\end{aligned}$$

We define the vector v as

$$\begin{aligned}
v \triangleq [&\hat{x}_1(y)-x_1,\ldots,\hat{x}_n(y)-x_n,\hat{\theta}_1(y)-\theta_1,\ldots,\hat{\theta}_m(y)-\theta_m,\\
&\nabla_{x_1}\log p(x,y|\theta),\ldots,\nabla_{x_n}\log p(x,y|\theta),\\
&\nabla_{\theta_1}\log p(x,y),\ldots,\nabla_{\theta_m}\log p(x,y|\theta)]^T, \qquad \text{(K.99)}
\end{aligned}$$

and the matrix $\mathbf{C}_v$ as

$$\mathbf{C}_v \triangleq \mathrm{E}_{XY|\Theta}[vv^T] = \begin{bmatrix} \mathbf{E}^{(X\Theta)}(\theta) & \mathbf{I} \\ \mathbf{I} & \mathbf{H}(x) \end{bmatrix}, \qquad \text{(K.100)}$$

where $\mathbf{I}$ is a unity matrix, and the RHS follows from (5.68), (5.69), (K.85), (K.90), (K.94), and (K.98); the matrix $\mathbf{C}_v$ is well-defined as a consequence of Assumption 1 and 2. We apply Lemma K.1, and obtain (5.77). □

The matrices $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ (cf. Example 5.9)

We compute the the matrices $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ (cf. (5.357) and (5.359)):

$$\begin{aligned}
\mathbf{G}^{(1)}_{11} &\triangleq \mathrm{E}_{XY|\mathbf{a},\sigma^2_W,\sigma^2_U}\left[\frac{\partial^2\log f_{1,k}}{\partial^2 x_k}\right] && \text{(K.101)}\\
&= -\frac{1}{\sigma^2_U} && \text{(K.102)}\\
\mathbf{G}^{(1)}_{i+1\,j+1} &= \mathrm{E}_{XY|\mathbf{a},\sigma^2_W,\sigma^2_U}\left[\frac{\partial^2\log f_{1,k}}{\partial x_{k-i}\partial x_{k-j}}\right] && \text{(K.103)}
\end{aligned}$$

$$= -\frac{a_i a_j}{\sigma_U^2} \quad (i,j=1,\ldots,M) \tag{K.104}$$

$$\mathbf{G}^{(1)}_{i+1\,1} = \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial x_{k-i}\partial x_k}\right] \tag{K.105}$$

$$= \frac{a_i}{\sigma_U^2} \quad (i=1,\ldots,M) \tag{K.106}$$

$$\mathbf{G}^{(1)}_{M+i\,M+j} \triangleq \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial a_i \partial a_j}\right] \tag{K.107}$$

$$= -\frac{1}{\sigma_U^2}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}[x_{k-i}x_{k-j}] \quad (i,j=1,\ldots,M) \tag{K.108}$$

$$\mathbf{G}^{(1)}_{2M+1\,2M+1} \triangleq \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial^2 \sigma_U^2}\right] \tag{K.109}$$

$$= \frac{1}{2\sigma_U^4} - \frac{1}{\sigma_U^6}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\left(x_k - \sum_{\ell=1}^{M} a_\ell x_{k-\ell}\right)^2\right] \tag{K.110}$$

$$= -\frac{1}{2\sigma_U^4} \tag{K.111}$$

$$\mathbf{G}^{(1)}_{M+i\,1+j} = \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial a_i \partial x_{k-j}}\right] \tag{K.112}$$

$$= -\frac{a_j}{\sigma_U^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}[x_{k-j}] \tag{K.113}$$

$$= 0 \quad (i,j=1,\ldots,M) \tag{K.114}$$

$$\mathbf{G}^{(1)}_{M+i\,1} = \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial a_i \partial x_k}\right] \tag{K.115}$$

$$= \frac{1}{\sigma_U^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}[x_{k-i}] \tag{K.116}$$

$$= 0 \quad (i=1,\ldots,M) \tag{K.117}$$

$$\mathbf{G}^{(1)}_{M+i\,2M+1} = \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial a_i \partial \sigma_U^2}\right] \tag{K.118}$$

$$= -\frac{1}{\sigma_U^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[x_{k-i}\left(x_k - \sum_{\ell=1}^{M} a_\ell x_{k-\ell}\right)\right] \tag{K.119}$$

$$= 0 \quad (i=1,\ldots,M) \tag{K.120}$$

$$\mathbf{G}^{(1)}_{i+1\,2M+1} = \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial x_{k-i}\partial \sigma_U^2}\right] \tag{K.121}$$

$$
\begin{aligned}
&= -\frac{a_i}{\sigma_U^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[x_k - \sum_{\ell=1}^{M} a_\ell x_{k-\ell}\right] && \text{(K.122)}\\
&= 0 \quad (i = 1, \ldots, M) && \text{(K.123)}\\
\mathbf{G}^{(1)}_{1\,2M+1} &= \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{1,k}}{\partial x_k \partial \sigma_U^2}\right] && \text{(K.124)}\\
&= \frac{1}{\sigma_U^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[x_k - \sum_{\ell=1}^{M} a_\ell x_{k-\ell}\right] && \text{(K.125)}\\
&= 0 && \text{(K.126)}\\
\mathbf{G}^{(2)}_{11} &\triangleq \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{2,k}}{\partial^2 x_k}\right] && \text{(K.127)}\\
&= -\frac{1}{\sigma_W^2} && \text{(K.128)}\\
\mathbf{G}^{(2)}_{2M+2,2M+2} &= \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{2,k}}{\partial^2 \sigma_W^2}\right] && \text{(K.129)}\\
&= \frac{1}{2\sigma_W^4} - \frac{1}{\sigma_W^6}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[(y_k - x_k)^2\right] && \text{(K.130)}\\
&= -\frac{1}{2\sigma_W^4} && \text{(K.131)}\\
\mathbf{G}^{(2)}_{1,2M+2} &= \mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[\frac{\partial^2 \log f_{2,k}}{\partial x_k \partial \sigma_W^2}\right] && \text{(K.132)}\\
&= -\frac{1}{\sigma_W^4}\mathrm{E}_{XY|\mathbf{a},\sigma_W^2,\sigma_U^2}\left[y_k - x_k\right] = 0. && \text{(K.133)}
\end{aligned}
$$

The other elements of $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ are zero.

Proof of Lemma 5.3

Note that

$$
p(y|\theta) \triangleq \int_x p(x, y|\theta)dx, \qquad \text{(K.134)}
$$

and therefore,

$$
\nabla_\theta p(y|\theta) = \nabla_\theta \int_x p(x, y|\theta)dx \qquad \text{(K.135)}
$$

$$
\begin{aligned}
&= \int_x \nabla_\theta p(x, y|\theta) dx && \text{(K.136)} \\
&= \int_x p(x, y|\theta) \nabla_\theta \log p(x, y|\theta) dx && \text{(K.137)} \\
&= p(y|\theta) \int_x p(x|\theta, y) \nabla_\theta \log p(x, y|\theta) dx. && \text{(K.138)}
\end{aligned}
$$

In (K.136), we made use of the assumption of differentiability under the integral sign.

As a consequence,

$$
\begin{aligned}
\nabla_\theta \log p(y|\theta) &= \nabla_\theta p(y|\theta)/p(y|\theta) && \text{(K.139)} \\
&= \int_x p(x|\theta, y) \nabla_\theta \log p(x, y|\theta) dx && \text{(K.140)} \\
&= \mathrm{E}_{X|\Theta Y} \left[\nabla_\theta \log p(X, y|\theta) \right]. && \text{(K.141)}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\mathrm{E}_{Y|\Theta} \left[\nabla_\theta \log p(Y|\theta) \nabla_\theta^T \log p(Y|\theta) \right] \\
&\quad = \mathrm{E}_{Y|\Theta} \left[\mathrm{E}_{X|\Theta Y} \left[\nabla_\theta \log p(X, Y|\theta) \right] \mathrm{E}_{X|\Theta Y} \left[\nabla_\theta \log p(X, Y|\theta) \right]^T \right]. \qquad \text{(K.142)}
\end{aligned}
$$

□

Proof of Lemma 5.4

Note that

$$
p(x, y) \triangleq \int_z p(x, z, y) dz, \qquad \text{(K.143)}
$$

and therefore,

$$
\begin{aligned}
\nabla_x p(x, y) &= \nabla_x \int_z p(x, z, y) dz && \text{(K.144)} \\
&= \int_z \nabla_x p(x, z, y) dz && \text{(K.145)} \\
&= \int_z p(x, z, y) \nabla_x \log p(x, z, y) dz && \text{(K.146)} \\
&= p(x, y) \int_z p(z|x, y) \nabla_x \log p(x, z, y) dz. && \text{(K.147)}
\end{aligned}
$$

In (K.145), we made use of the assumption of differentiability under the integral sign.

As a consequence,

$$\begin{aligned}\nabla_x \log p(x,y) &= \nabla_x p(x,y)/p(x,y) && \text{(K.148)}\\ &= \int_z p(z|x,y)\nabla_x \log p(x,z,y)dz && \text{(K.149)}\\ &= \mathrm{E}_{Z|XY}\left[\nabla_x \log p(x,Z,y)\right]. && \text{(K.150)}\end{aligned}$$

Therefore,

$$\begin{aligned}&\mathrm{E}_{XY}\left[\nabla_x \log p(X,Y)\nabla_x^T \log p(X,Y)\right]\\ &\quad= \mathrm{E}_{XY}\left[\mathrm{E}_{Z|XY}\left[\nabla_x \log p(X,Y,Z)\right]\mathrm{E}_{Z|XY}\left[\nabla_x \log p(X,Y,Z)\right]^T\right]. \qquad \text{(K.151)}\end{aligned}$$

□

CRB for estimation in AR model: CRB for Θ (cf. Example 5.10)

The components of (the vectors) $\mathrm{E}_{X|\Theta Y}[\nabla_\theta \log f_{1,k}]$ and $\mathrm{E}_{X|\Theta Y}[\nabla_\theta \log f_{2,k}]$ (with $\Theta \overset{\triangle}{=} (a_1,\ldots,a_M,\sigma_U^2,\sigma_W^2)$, cf. (5.419)) can be written as :

$$\begin{aligned}&\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\big[\nabla_{a_i}\log f_1(X_k,\ldots,X_{k-M},\mathbf{a},\sigma_U^2)\big]\\ &\quad= \frac{1}{\sigma_U^2}\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[X_{k-i}\big(X_k - \sum_{\ell=1}^{M} a_\ell X_{k-\ell}\big)\Big] \qquad \text{(K.152)}\\ &\quad= \frac{1}{\sigma_U^2}\big(\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\big[X_{k-i}X_k\big] - \sum_{\ell=1}^{M} a_\ell \mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\big[X_{k-i}X_{k-\ell}\big]\big) \qquad \text{(K.153)}\end{aligned}$$

$$\begin{aligned}&\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[\nabla_{\sigma_U^2}\log f_1(X_k,\ldots,X_{k-M},\mathbf{a},\sigma_U^2)\Big]\\ &\quad= -\frac{1}{2\sigma_U^2} + \frac{1}{2\sigma_U^4}\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[\big(X_k - \sum_{\ell=1}^{M} a_\ell X_{k-\ell}\big)^2\Big] \qquad \text{(K.154)}\\ &\quad= -\frac{1}{2\sigma_U^2} + \frac{1}{2\sigma_U^4}\Big(\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\big[X_k^2\big] - 2\sum_{\ell=1}^{M} a_\ell \mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\big[X_k X_{k-\ell}\big]\end{aligned}$$

$$+\sum_{\ell=1}^{M}\sum_{m=1}^{M} a_\ell a_m \mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[X_{k-\ell}X_{k-m}\Big]\Big) \tag{K.155}$$

$$\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[\nabla_{\sigma_W^2}\log f_2(X_k,\sigma_W^2,y_k)\Big]$$
$$= -\frac{1}{2\sigma_W^2}+\frac{1}{2\sigma_W^4}\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\left[(y_k-X_k)^2\right] \tag{K.156}$$
$$= -\frac{1}{2\sigma_W^2}+\frac{1}{2\sigma_W^4}\Big(y_k^2-2y_k\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}[X_k]+\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\left[X_k^2\right]\Big). \tag{K.157}$$

The other components are zero, e.g.,

$$\mathrm{E}_{X|\mathbf{a}\,\sigma_W^2\sigma_U^2 Y}\Big[\nabla_{\sigma_U^2}\log f_2(X_k,\sigma_W^2,y_k)\Big]=0. \tag{K.158}$$

In Appendix J.6, we explain how the correlations $\mathrm{E}\big[X_iX_j\big]$ can be computed.

CRB for estimation in AR model: hybrid CRB for X_k (cf. Example 5.10)

We derive a hybrid CRB for the variable X_k $(k=1,\ldots,N)$ from the marginal $p(x_k,y|\mathbf{a},\sigma_W^2,\sigma_U^2)$. The key is Lemma 5.6 with $X\triangleq X_k$, $Z=X_{\bar{k}}\triangleq(X_1,\ldots,X_{k-1},X_{k+1},\ldots,X_N)$, and $\Theta\triangleq(\mathbf{a},\sigma_W^2,\sigma_U^2)$; the expressions (5.402)–(5.404) can be evaluated along the lines of (5.372). In the following, we explain how the expectations $\mathrm{E}_{Z|XY\Theta}[\nabla_\theta\log p(\cdot)]$ and $\mathrm{E}_{Z|XY\Theta}[\nabla_x\log p(\cdot)]$ in (5.402)–(5.404) may be determined. The components of $\mathrm{E}_{Z|XY\Theta}[\nabla_\theta\log p(\cdot)]$ are similar to (5.419) and (K.152)–(K.157): one needs to replace the expectations over $p(x|\mathbf{a},\sigma_W^2,\sigma_U^2,Y)$ by expectations over $p(x_{\bar{k}}|x_k,\mathbf{a},\sigma_W^2,\sigma_U^2,Y)$. We have:

$$\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\left[\nabla_\theta\log p(\theta,x_k,X_{k-1}\ldots,X_{k-M},y)\right]$$
$$=\sum_{k=1}^{N}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\left[\nabla_\theta\log f_1(x_k,X_{k-1}\ldots,X_{k-M},\mathbf{a},\sigma_U^2)\right]$$
$$+\sum_{k=1}^{N}\nabla_\theta\log f_2(x_k,y_k,\sigma_W^2), \tag{K.159}$$

and,

$$\begin{aligned}
&\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[\nabla_{a_i}\log f_1(x_k, X_{k+1}\ldots, x_{k-M}, \mathbf{a}, \sigma_U^2)\big] \\
&\quad = \frac{1}{\sigma_U^2}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\Big[X_{k-i}\big(x_k - \sum_{\ell=1}^{M} a_\ell X_{k-\ell}\big)\Big] \qquad \text{(K.160)} \\
&\quad = \frac{x_k}{\sigma_U^2}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[X_{k-i}\big] \\
&\qquad\quad - \frac{1}{\sigma_U^2}\sum_{\ell=1}^{M} a_\ell \mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\Big[X_{k-i}X_{k-\ell}\Big] \qquad \text{(K.161)} \\
&\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\Big[\nabla_{\sigma_U^2}\log f_1(X_k,\ldots, X_{k-M}, \mathbf{a}, \sigma_U^2)\Big] \\
&\quad = -\frac{1}{2\sigma_U^2} + \frac{1}{2\sigma_U^4}\Big(x_k^2 - 2\sum_{\ell=1}^{M} a_\ell x_k \mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[X_{k-\ell}\big] \\
&\qquad\qquad + \sum_{\ell=1}^{M}\sum_{m=1}^{M} a_\ell a_m \mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\Big[X_{k-\ell}X_{k-m}\Big]\Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(K.162)} \\
&\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\Big[\nabla_{\sigma_W^2}\log f_2(X_k, \sigma_W^2, y_k)\Big] \\
&\quad = -\frac{1}{2\sigma_W^2} + \frac{1}{2\sigma_W^4}(y_k - x_k)^2. \qquad \text{(K.163)}
\end{aligned}$$

Along similar lines, one obtains $\mathrm{E}_{Z|XY\Theta}[\nabla_x \log p(x, Z, y|\theta)]$:

$$\begin{aligned}
&\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\left[\nabla_{x_k}\log p(\theta, x_k, X_{k-1},\ldots, X_{k-M}, y)\right] \\
&= \sum_{m=0}^{M}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[\nabla_{x_k}\log f_1(x_{k+m}, X_{k-1+m}\ldots, X_{k-M+m}, \mathbf{a}, \sigma_U^2)\big] \\
&\qquad\qquad + \nabla_{x_k}\log f_2(x_k, y_k, \sigma_W^2), \qquad \text{(K.164)}
\end{aligned}$$

where:

$$\begin{aligned}
&\sum_{m=0}^{M}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[\nabla_{x_k}\log f_1(x_{k+m}, X_{k-1+m}\ldots, X_{k-M+m}, \mathbf{a}, \sigma_U^2)\big] \\
&\quad = -\frac{x_k}{\sigma_U^2} + \sum_{\ell=1}^{M}\frac{a_\ell}{\sigma_U^2}\mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[X_{k-\ell}\big] \\
&\qquad + \frac{1}{\sigma_U^2}\sum_{m=1}^{M} a_m \mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[X_{k+m}\big]
\end{aligned}$$

$$- \sum_{m=1}^{M} \sum_{\ell=1}^{M} \frac{a_\ell a_m}{\sigma_U^2} \mathrm{E}_{X_{\bar{k}}|X_k,\mathbf{a},\sigma_W^2,\sigma_U^2 Y}\big[X_{k-\ell+m}\big], \tag{K.165}$$

and

$$\nabla_{x_k} \log f_2(x_k, y_k, \sigma_W^2) = \frac{y_k - x_k}{\sigma_W^2}. \tag{K.166}$$

In Appendix J.6, we explain how the correlations $\mathrm{E}\big[X_i X_j\big]$ can be computed.

Appendix L

Alternative Message Update Rules for the Soft LFSR

For the convenience of the reader, we explicitly state all computations in the soft LFSR for an alternative (more standard) version of the sum-product algorithm, for the max-product (min-sum) algorithm, as well as for the analog LFSR of Gershenfeld and Grinstein.

L.1 Sum-Product LFSR for Likelihood Ratio Representation

If the messages represent the *ratio* $\tilde{p}(0)/\tilde{p}(1)$ of the pseudo-probabilities, the sum-product update rules of the soft LFSR are as follows.

Initialization: $\mu_k = 1$ for $k = -m+1, -m+2, \ldots, 0$.

Recursion (for $k = 1, 2, 3, \ldots$):

$$\mu_{A,k} \quad = \quad \frac{p(y_k|x_k = 0)}{p(y_k|x_k = 1)} \tag{L.1}$$

$$\overset{\text{for AWGN}}{=} \quad \exp(2y_k/\sigma^2) \tag{L.2}$$

$$\mu_{B,k} = \frac{1 + \mu_{k-\ell} \cdot \mu_{k-m}}{\mu_{k-\ell} + \mu_{k-m}} \tag{L.3}$$

$$\mu_k = \mu_{A,k} \cdot \mu_{B,k} \tag{L.4}$$

At any given time k, an estimate of X_k is obtained as

$$\hat{X}_k \triangleq \begin{cases} 0, & \text{if } \mu_k \geq 1 \\ 1, & \text{if } \mu_k < 1 \end{cases} \tag{L.5}$$

and $[\hat{X}_k] = (\hat{X}_{k-m+1}, \ldots, \hat{X}_{k-1}, \hat{X}_k)$ is an estimate of the state $[X_k]$.

L.2 Max-Product (Max-Sum) Soft LFSR

We state the max-product soft LFSR [103], [119] for the case where the messages represent $\ln(\tilde{p}(0)/\tilde{p}(1))$.

Initialization: $\mu_k = 0$ for $k = -m+1, -m+2, \ldots, 0$.

Recursion (for $k = 1, 2, 3, \ldots$):

$$\mu_{A,k} = \ln \frac{p(y_k|x_k = 0)}{p(y_k|x_k = 1)} \tag{L.6}$$

$$\overset{\text{for AWGN}}{=} \quad 2y_k/\sigma^2 \tag{L.7}$$

$$|\mu_{B,k}| = \min\{|\mu_{k-\ell}|, |\mu_{k-m}|\} \tag{L.8}$$

$$\operatorname{sgn}(\mu_{B,k}) = \operatorname{sgn}(\mu_{k-\ell}) \cdot \operatorname{sgn}(\mu_{k-m}) \tag{L.9}$$

$$\mu_k = \mu_{A,k} + \mu_{B,k} \tag{L.10}$$

where $\operatorname{sgn}(x)$ denotes the sign of x. Finally, we have

$$\hat{X}_k \triangleq \begin{cases} 0, & \text{if } \mu_k \geq 0 \\ 1, & \text{if } \mu_k < 0 \end{cases} \tag{L.11}$$

In fact, (L.7) may be replaced by

$$\mu_{A,k} = y_k, \tag{L.12}$$

which amounts to multiplying all messages by $\sigma^2/2$ and does not change the estimate (L.11).

L.3 Analog LFSR by Gershenfeld and Grinstein

In [74], Gershenfeld and Grinstein obtained a discrete-time "analog" LFSR by embedding the discrete dynamics of the LFSR into a continuous state space. They showed that such an analog LFSR entrains to a LFSR sequence even if the latter is modulated by a weak data signal. An extension of this approach to continuous time (using ideal continuous-time delay cells) is also given in [74]. In the setup of Chapter 8, the analog LFSR of [74] can be described as follows.

Initialization: $\mu_k = 0$ for $k = -m+1, -m+2, \ldots, 0$.

Recursion (for $k = 1, 2, 3, \ldots$):

$$\mu_{A,k} = y_k \tag{L.13}$$

$$\mu_{B,k} = \cos\left[\pi\left(\frac{1-\mu_{k-\ell}}{2} + \frac{1-\mu_{k-m}}{2}\right)\right] \tag{L.14}$$

$$\mu_k = (1-\epsilon)\,\mu_{B,k} + \epsilon\,\mu_{A,k} \tag{L.15}$$

or, alternatively,

$$\mu_k = \begin{cases} \mu_{B,k} & \text{if } ||\mu_{A,k}| - 1| > \delta \\ (1-\epsilon)\,\mu_{B,k} + \epsilon\,\mathrm{sgn}(\mu_{A,k}) & \text{otherwise} \end{cases} \tag{L.16}$$

and

$$\hat{X}_k \triangleq \begin{cases} 0, & \text{if } \mu_k \geq 0 \\ 1, & \text{if } \mu_k < 0 \end{cases} \tag{L.17}$$

In this formulation (and differing from [74]), the "hard" logical values 0 and 1 are represented as $+1$ and -1, respectively. It should be noted that [74] does not explicitly consider noise at all.

In our simulations, we used (L.16) with $\delta = \infty$ and optimized ϵ (≈ 0.4 for large SNR).

Abbreviations

AR	Auto-Regression
AWGN	Additive White Gaussian Noise
BCRB	Bayesian Cramér-Rao Bound
CLT	Central Limit Theorem
CRB	Cramér-Rao Bound
EM	Expectation Maximisation
GEM	Gradient EM
HCRB	Hybrid Cramér-Rao Bound
HEM	Hybrid EM
HMM	Hidden Markov Model
ICM	Iterative Conditional Modes
i.i.d.	independent identically distributed
LFSR	Linear Feedback Shift Register
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
ML	Maximum-Likelihood
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
pdf	probability density function
pmf	probability mass function
SA	Stochastic Approximation
SNR	Signal-to-Noise Ratio
SPA	Summary-Propagation Algorithm
w.r.t.	with respect to

List of Symbols

Elementary

X	Random variable (or vector)
x	Value of random variable (or vector) X
$\hat{x}$	Estimate of variable (or parameter or vector) x
$\mathbf{X}$	Random vector
$\mathbf{x}$	Value of random vector $\mathbf{X}$
$\hat{\mathbf{x}}$	Estimate of vector (or parameter) $\mathbf{x}$
$\mathbf{A}$	Matrix
$\triangleq$	Definition
$\propto$	Proportional to
$\overset{!}{=}$	Set to equality

Algebra

$\mathbb{Z}$	Ring of integers
$\mathbb{R}$	Field of real numbers
$\mathbb{R}^+$	Field of positive real numbers
$\mathbb{R}_0^+$	Field of non-negative real numbers
$\mathbb{C}$	Field of complex numbers

Linear Algebra

$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^H$	Hermitian transpose of matrix $\mathbf{A}$
$\mathrm{diag}\,(\cdots)$	Diagonal matrix
$\mathrm{Trace}(\cdot)$	Trace operator
$\mathbf{I}$	Identity matrix

Probability

$\mathcal{N}(\,.\mid\mu,\sigma^2)$	Scalar Gaussian distribution with mean μ and variance σ^2
$\mathcal{N}(\,.\mid\mathbf{m},\mathbf{V})$	Multivariate Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{V}$
$\mathcal{N}^{-1}(\,.\mid\mathbf{m},\mathbf{W})$	Multivariate Gaussian distribution with mean vector $\mathbf{m}$ and weight matrix $\mathbf{W}$
$\mathbf{W}$	Weight matrix of a Gaussian distribution
$\mathbf{V}$	Covariance matrix of a Gaussian distribution
$\mathbf{m}$	Mean vector of a Gaussian distribution
$\mathrm{Ig}\,(\,.\mid\alpha,\beta)$	Inverted-gamma distribution with parameters α, β
$\mathrm{E}[X]$	Expectation of r.v. X
$\mathrm{Var}[X]$	Variance of r.v. X
$\mathrm{M}[X]$	Mode (i.e. maximum) of the distribution of r.v. X

Factor graph

$\mu_{f\to X}(x)$	Message leaving node f along edge X
$\mu_{X\to f}(x)$	Message arriving at node f along edge X
$h(\theta)$	Local EM message

Miscellaneous

$\hat{\theta}^{(k)}$	estimate of θ in the k-th iteration
$[\mathbf{x}]_i$	i-th element of vector $\mathbf{x}$
$[\mathbf{A}]_{ij}$	Element of matrix $\mathbf{A}$ in row i and column j

Bibliography

[1] Collection of papers on "Capacity Approaching Codes, Iterative Decoding Algorithms, and Their Applications" in *IEEE Communications Mag.,* vol. 41, August 2003.

[2] Special Issue on "Codes on Graphs and Iterative Decoding" of *IEEE Trans. Information Theory,* vol. 47, Feb. 2001.

[3] A. Abel and W. Schwarz, "Chaos Communications—Principles, Schemes, and System Analysis," *Proc. IEEE,* vol. 90, pp. 691–710, May 2002.

[4] I. C. Abou-Faycal, M. D. Trott, and S. Shamai, "The Capacity of Discrete-Time Memoryless Rayleigh-Fading Channels," *IEEE Trans. Information Theory,* vol. 47, no. 4, pp. 1290–1301, May 2001.

[5] S. M. Aji, G. B. Horn, and R. J. McEliece, "Iterative Decoding on Graphs with a Single Cycle," in *Proc. IEEE Intern. Symp. on Inform. Theory,* MIT, Cambridge, MA, USA, Aug. 16–21 1998, p. 276.

[6] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Information Theory*, vol. 46, no. 2, pp. 325–343, 2000.

[7] S. I. Amari and H. Nagaoka, *Methods of Information Geometry, AMS Translations of Mathematical Monographs,* vol. 191, American Mathematical Society (AMS) and Oxford University Press, 2000.

[8] S. I. Amari and S. C. Douglas, "Why Natural Gradient?", *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98),* Seattle, WA, USA, 12–15 May 1998, vol. 2 pp. 1213–1216.

[9] S. Arimoto, "An Algorithm for Computing the Capacity of Arbitrary Discrete Memoryless Channels," *IEEE Trans. Information Theory*, vol. 18, pp. 14–20, 1972.

[10] D. Arnold, *Computing Information Rates of Finite-State Models with Application to Magnetic Recording,* ETH Dissertation no. 14760, ETH Zurich, Switzerland, 2002.

[11] D. Arnold, A. Kavčić, H.-A. Loeliger, P. O. Vontobel, and W. Zeng, "Simulation-Based Computation of Information Rates: Upper and Lower Bounds," *Proc. 2003 IEEE Int. Symp. Information Theory,* Yokohama, Japan, June 29–July 4, 2002, p. 119.

[12] D. Arnold and H.-A. Loeliger, "On the Information Rate of Binary-Input Channels with Memory," *Proc. 2001 IEEE Int. Conf. on Communications,* Helsinki, Finland, June 11–14, 2001, pp. 2692–2695.

[13] D. Arnold, H.-A. Loeliger, and P. O. Vontobel, "Computation of Information Rates from Finite-State Source/Channel Models," *Proc. 40th Annual Allerton Conference on Communication, Control, and Computing,* (Allerton House, Monticello, Illinois), October 2–4, 2002, pp. 457–466.

[14] A. B. Baggeroer and H. Schmidt, "Cramér-Rao Bounds for Matched Field Tomography and Ocean Acoustic Tomography," *Proc. 1995 International Conference on Acoustics, Speech, and Signal Processing,* Detroit, Michigan, USA, May 9–12, 1995, vol. 5, pp. 2763–2766.

[15] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Information Theory,* vol. 20, pp. 284–287, March 1974.

[16] R. Bamler, "Doppler Frequency Estimation and the Cramér-Rao bound," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 29, May 1991, pp. 385–390.

[17] A. Barron, "The Strong Ergodic Theorem for Densities: Generalized Shannon-McMillan-Breiman Theorem," *Annals of Prob.*, vol. 13, no. 4, pp. 1292–1303, 1995.

[18] S. C. Beaulieu, "On the Application of the Cramér-Rao and Detection Theory Bounds to Mean Square Error of Symbol Timing Recovery," *IEEE Transactions on Communications,* vol. 40, Oct. 1992, pp. 1635–1643.

[19] D. P. Bertsekas, *Nonlinear Programming,* Athena Scientific, Belmont, MA, 1995.

[20] C. Berzuini, N. G. Best, W. R. Gilks, and C. Larizza, "Dynamic Conditional Independence Models and Markov-Chain-Monte-Carlo Methods", *Journal of the American Statistical Association,* vol. 92, no. 440, pp. 1403–1412, 1997.

[21] A. Bhattacharyya, "On Some Analogues of the Amount of Information and Their Use in Statistical Estimation," *SANHKYA,* vol. 8, pp. 315–328, 1948.

[22] C. M. Bishop, *Neural Networks for Pattern Recognition,* Oxford University Press, 1995.

[23] R. Blahut, "Computation of Channel Capacity and Rate-Distortion Functions," *IEEE Trans. Information Theory*, vol. 18, pp. 460–473, 1972.

[24] B. Bobrovsky, E. Mayer-Wolf, and M. Zakai, "Some Classes of Global Cramér-Rao Bounds," *A. Statistics*, vol. 15, pp. 1421–1438, 1987.

[25] B. Z. Bobrovsky and M. Zakai, "A Lower Bound on the Estimation Error for Certain Diffusion Processes," *IEEE Trans. Information Theory,* vol. 22, pp. 45–52, Jan. 1976.

[26] M. G. S. Bruno and A. Pavlov, "Improved Particle Filters for Ballistic Target Tracking," *Proc. 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing,* Montreal, Quebec, Canada, May 17–21, 2004, vol. 2, pp. 705–708.

[27] A. Burr and L. Zhang, "Application of Turbo-Principle to Carrier Phase Recovery in Turbo Encoded Bit-Interleaved Coded Modulation System," *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, Brest, France, 1–5 Sept., 2003, pp. 87–90.

[28] T. Cassaro and C. N. Georghiades, "Carrier and Phase Recovery for Coded Systems Using a Preamble Sequence with Reliability Information over AWGN Channels," *IEEE International Conference on Communications (ICC 2000),* vol. 1, pp. 326–330, June 2000.

[29] M. Chakraborty and S. Prasad, "Computation of a Useful Cramér-Rao Bound for Multichannel ARMA Parameter Estimation," *IEEE Transactions on Signal Processing,* vol. 42, no. 2, Feb. 1994, pp. 466–469.

[30] T. Chan, S. Hranilovic, and F. Kschischang, "Capacity-Achieving Probability Measure for Conditionally Gaussian Channels with Bounded Inputs," submitted to IEEE Trans. Information Theory, Dec. 2004.

[31] C. Chang and L. D. Davisson, "On Calculating the Capacity of an Infinite-Input Finite (Infinite)-Output Channel," *IEEE Trans. Information Theory,* vol. 34, Sept. 1998, pp. 1004–1010.

[32] K. M. Chugg and M. Zhu, "A New Approach to Rapid PN Code Acquisition Using Iterative Message Passing Techniques," *IEEE J. Select. Areas Comm.,* vol. 23, pp. 884–897, May 2005.

[33] C. K. Chui and G. Chen, *Kalman filtering with Real-Time Applications*, Springer Series in Information Sciences, 1990.

[34] A. Cichocki and S. I. Amari, *Adaptive Blind Signal and Image Processing,* John Wiley and Sons, 2002.

[35] G. Colavolpe, G. Ferrari, and R. Raheli, "Noncoherent Iterative (Turbo)Decoding," *IEEE Trans. Comm.* vol. 48, no. 9, pp. 1488–1498, Sept. 2000.

[36] G. Colavolpe and G. Caire, "Iterative Decoding in the Presence of Strong Phase Noise," *IEEE Journal on Selected Areas in Communications,* Differential and Noncoherent Wireless Communications, to appear.

[37] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: Wiley, 1991.

[38] K. M. Cuomo and A. V. Oppenheim, "Circuit Implementation of Synchronized Chaos with Applications to Communications," *Physical Review Letters,* vol. 71, July 1993.

[39] J. Dauwels, "Numerical Computation of the Capacity of Continuous Memoryless Channels," *Proc. of the 26th Symposium on Information Theory in the BENELUX,* 2005, to appear.

[40] J. Dauwels, "Computing Bayesian Cramér-Rao bounds," *Proc. of the IEEE Int. Symp. Information Theory,* 2005, to appear.

[41] J. Dauwels, "A Numerical Method to Compute Cramér-Rao-Type Bounds for Challenging Estimation Problems," *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* 2006, to appear.

[42] J. Dauwels, M. Frey, T. Koch, H.-A. Loeliger, P. Merkli, and B. Vigoda, *Synchronization of a Pseudo-Noise Signal using an Analog Circuit,* Technical Report No. 200401, Signal and Information Processing Laboratory, ETH Zurich, Switzerland, January 2004.

[43] J. Dauwels, M. Frey, T. Koch, H.-A. Loeliger, P. Merkli, and B. Vigoda, *An Analog Circuit that Locks onto a Pseudo-Noise Signal,* Technical Report No. 200403, Signal and Information Processing Laboratory, ETH Zurich, Switzerland, June 2004.

[44] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation Maximization for Phase Estimation," *Proc. of the Eighth International Symposium on Communication Theory and Applications,* 2005, to appear.

[45] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation Maximization as Message Passing", *Proc. 2005 IEEE Int. Symp. Information Theory,* to appear.

[46] J. Dauwels, S. Korl, and H.-A. Loeliger, "Steepest Descent on Factor Graphs", *Proc. 2005 IEEE ITSOC Information Theory Workshop on Coding and Complexity*, to appear.

[47] J. Dauwels and H.-A. Loeliger,"Joint Decoding and Phase Estimation: an Exercise in Factor Graphs," *Proc. 2003 IEEE Int. Symp. Information Theory,* p. 231, Yokohama, Japan, June 29–July 4, 2003.

[48] J. Dauwels and H. -A. Loeliger, "Phase Estimation by Message Passing," *IEEE International Conference on Communications*, Paris, France, June 20–24, 2004, pp. 523–527.

[49] J. Dauwels and H.-A. Loeliger, "Computation of Information Rates by Particle Methods," *Proc. 2004 IEEE International Symposium on Information Theory,* p. 178, Chicago, USA , June 27–July 2, 2004.

[50] J. Dauwels, H.-A. Loeliger, P. Merkli, and M. Ostojic, "On Structured-Summary Propagation, LFSR Synchronization, and Low-Complexity Trellis Decoding," *Proc. 41st Allerton Conf. on Communication, Control, and Computing, (Allerton House, Monticello, Illinois),* Oct. 1–3, 2003.

[51] J. Dauwels, H.-A. Loeliger, P. Merkli, and M. Ostojic, "On Markov structured Summary Propagation and LFSR Synchronization," *Proc. 42nd Allerton Conf. on Communication, Control, and Computing, (Allerton House, Monticello, Illinois),* Sept. 29–Oct. 1, 2004.

[52] J. Dauwels, H. Wymeersch, H.-A. Loeliger, and M. Moeneclaey, "Phase Estimation and Phase Ambiguity Resolution by Message Passing," *Proc. 11th International Conference on Telecommunications and Networking,* pp. 150–155, Fortaleza, Brazil, August 1–6, 2004.

[53] A. Demir and J. Roychowdhury, "On the Validity of Orthogonally Decomposed Perturbations in Phase Noise Analysis," Technical Report, Lucent Technologies, July 1997.

[54] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterisation," *IEEE Trans. Circuits Syst. I,* vol. 47, pp. 655–674, May 2000.

[55] A. Demir, "Floquet Theory and Nonlinear Perturbation Analysis for Oscillators with Differential-Algebraic Equations," *Int. J. of Circ. Theor. Appl.,* vol. 28, pp. 163–185, 2000.

[56] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterization" *IEEE Trans. Circuits Syst. I,* vol. 47, no. 5, May 2000, pp. 655–674.

[57] A. Demir, "Phase Noise and Timing Jitter in Oscillators with Colored Noise Sources," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications,* vol. 49, no. 12, December 2002.

[58] A. P. Dempster, N. M. Laird, and D. B. Rubin "Maximum Likelihood from Incomplete Data via the EM Algorithm,"*Journal of the Royal Statistical Society,* B 39, pp. 1–38, 1977.

[59] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, eds., *Sequential Monte Carlo Methods in Practice.* New York: Springer-Verlag, 2001.

[60] R. L. Dobrušin, "General Formulation of Shannon's Main Theorem in Information Theory," in *American Mathematical Society Translations,* ser. 2, 1963, vol. 33, pp. 323–438.

[61] A. W. Eckford and S. Pasupathy, "Iterative Multiuser Detection with Graphical Modeling" *IEEE International Conference on Personal Wireless Communications,* Hyderabad, India, 2000.

[62] S. Egner, V. B. Balakirsky, L. Tolhuizen, S. Baggen, and H. Hollmann, "On the Entropy Rate of a Hidden Markov Model," *Proc. 2004 IEEE*

International Symposium on Information Theory, p. 12, Chicago, USA , June 27–July 2, 2004.

[63] R. Fisher, "The Logic of Inductive Inference," *Journal of the Royal Society of Statistics,* vol. 98, pp. 39–54, 1935.

[64] J. Foo and R. J. Weber, "Low Power 5GHz Quadrature Phase CMOS LC Oscillator with Active Inductor," *Proc. IEEE International Symposium on Communications and Information Technology, (ISCIT 2004),* vol. 2, pp. 1084–1089, 26–29 Oct. 2004.

[65] G. D. Forney Jr., "The Viterbi Algorithm," *Proc. IEEE,* vol. 61, pp. 268–278, March 1973.

[66] G. D. Forney Jr., "Codes on Graphs: Normal Realizations," *IEEE Trans. Information Theory,* vol. 47, no. 2, pp. 520–548, 2001.

[67] L. Frenkel and M. Feder, "Recursive Estimate-Maximize (EM) Algorithms for Time-Varying Parameters with Applications to Multiple Target Tracking," *IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 3, 9–12 May, 1995, pp. 2068–2071.

[68] B. Frey, *Graphical Models for Machine Learning and Digital Communication,* MIT Adaptive Computation And Machine Learning Series, 1998.

[69] B. Frey and N. Jojic, *A Comparison of Algorithms for Inference and Learning in Probabilistic Graphical Models*, Technical Report PSI-2003-22, University of Toronto, 2003.

[70] B. Friedlander, "On the Computation of the Cramér-Rao Bound for ARMA Parameter Estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 32, no. 4, August 1984, pp. 721–727.

[71] B. Friedlander and B. Porat, "The Exact Cramér-Rao Bound for Gaussian Autoregressive Processes," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 25, Jan. 1989, pp. 3–7.

[72] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[73] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963.

[74] N. Gershenfeld and G. Grinstein, "Entrainment and Communication with Dissipative Pseudorandom Dynamics," *Physical Review Letters,* vol. 74, pp. 5024–5027, June 1995.

[75] R. Gill and B. Levit, "Applications of the Van Trees Inequality: a Bayesian Cramér-Rao Bound," *Bernoulli,* vol. 1, 1995, pp. 59–79.

[76] M. Ghogho, A. Nandi, and A. Swami, "Cramér-Rao Bounds and Parameter Estimation for Random Amplitude Phase Modulated Signals", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '99),* vol. 3, March 15–19, 1999 pp. 1577–1580.

[77] E. Hafner, "The Effects of Noise in Oscillators", *Proc. of the IEEE,* vol. 54, no. 2, p. 179, 1966.

[78] A. Hajimiri and T. Lee, *The Design of Low Noise Oscillators,* Springer, 1999.

[79] A. Handzel, T. Grossman, E. Domany, S. Tarem, and E. Duchovni, "A Neural Network Classifier in Experimental Particle Physics," *Int. J. Neural Syst.,* vol. 4, pp. 95–108, 1993.

[80] A. J. Hartemink, "Reverse Engineering Gene Regulatory Networks," *Nature Biotechnology,* vol. 23, pp. 554–555, 2005.

[81] F. Herzel and B. Razavi, "A Study of Oscillator Jitter Due to Supply and Substrate Noise," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 46, no. 1, Jan. 1999, p. 56–62.

[82] C. Herzet, V. Ramon, and L. Vandendorpe, "Turbo Synchronization: A Combined Sum-Product and Expectation-Maximization Algorithm Approach," *SPAWC'05 - IEEE Workshop on Signal Processing Advances in Wireless Communications,* June 5–8, 2005, New-York, USA.

[83] T. Heskes, "Stable Fixed Points of Loopy Belief Propagation are Minima of the Bethe Free Energy," *Proc. Neural Information Processing Systems (NIPS),* vol. 15, pp. 343–350, 2003.

[84] D. J. Higham, "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations," *SIAM Review, Society for Industrial and Applied Mathematics,* vol. 43, no. 3, pp. 525–546, 2001.

[85] H. Holzman, *Some Remarks on the Central Limit Theorem for Stationary Markov Processes*, PhD. Thesis, University of Göttingen, 1994.

[86] A. Host-Madsen and P. Handel, "Effects of Sampling and Quantization on Single-Tone Frequency Estimation," *IEEE Transactions on Signal Processing,* vol. 48, pp. 650–662, March 2000.

[87] Q. Huang, "Phase Noise to Carrier Ratio in LC Oscillators" *IEEE Trans. Circuits Syst. I,* vol. 47, no. 7., Jul. 2000, pp. 965–980.

[88] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient Multiscale Sampling from Products of Gaussian Mixtures," *NIPS,* Dec. 2003.

[89] T. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers," *In Advances in Neural Information Processing Systems,* vol. 11, 1998.

[90] T. Jaakkola and M. Jordan, "Improving the Mean Field Approximation via the Use of Mixture Distributions," *Proceedings of the NATO ASI on Learning in Graphical Models,* Kluwer, 1997.

[91] T. Jebara, R. Kondor, and A. Howard, "Probability Product Kernels" *Journal of Machine Learning Research, JMLR, Special Topic on Learning Theory,* no. 5, pp. 819–844, 2004.

[92] J. Johnson, “Thermal Agitation of Electricity in Conductors”, Phys. Rev. 32, 97 (1928).

[93] *Learning in Graphical Models,* Editor M. I. Jordan, MIT Press Cambridge, MA, USA, 1999.

[94] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, *An Introduction to Variational Methods for Graphical Models,* Machine Learning, vol. 37, no. 2, 1999.

[95] B. Kaulakys, “Modeling 1/f Noise,” Physics Letters E vol. 58 no. 6 (1998) 7013–7019.

[96] B. Kaulakys, “Stochastic Nonlinear Differential Equation Generating 1/f Noise,” Physics Letters, A 257 (2004) 37–42.

[97] Z. Khan, T. Balch, and F. Dellaert, “MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Accepted for publication, to appear in 2005.

[98] J. J. Kim, Y. Lee, and S. B. Park, “Low-Noise CMOS LC Oscillator with Dual-Ring Structure,” *Electronics Letters,* vol. 40, no. 17, pp. 1031–1032, Aug. 2004.

[99] T. Koch, *Continuous-Time Synchronization,* Semester Project, Signal and Information Processing Laboratory, ETH Zurich, Switzerland, July 2003.

[100] S. Korl, *A Factor Graph Approach to Signal Modelling, System Identification, and Filtering,* PhD. Thesis at ETH Zurich, Diss. ETH No 16170, July 2005.

[101] S. Korl, H.-A. Loeliger, and A.-G. Lindgren, “AR Model Parameter Estimation: From Factor Graphs to Algorithms,” *Proc. ICASSP’04,* vol. 5, Montreal, Canada, May 17–21, 2004, pp. 509–512.

[102] V. Krishnamurthy and J.B. Moore, “On-line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure,” *IEEE Transactions on Signal Processing,* vol. 41, Aug. 1993, pp. 2557–2573.

[103] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor Graphs and the Sum-Product Algorithm,” *IEEE Trans. Information Theory,* vol. 47, pp. 498–519, Feb. 2001.

[104] K. Kurokawa, “Noise in Synchronized Oscillators”, *IEEE Trans. on Microwave Theory and Techniques,* MTT vol. 16, pp. 234–240, 1968.

[105] J. D. Lafferty and L. A. Wasserman, “Iterative Markov Chain Monte-Carlo computation of Reference Priors and Minimax Risk,” *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence,* August 02–05, 2001, pp. 293–300.

[106] S. Lang, *Undergraduate Analysis*, Springer Verlag, 1997.

[107] E. K. Larsson and E. G. Larsson, “The CRB for Parameter Estimation in Irregularly Sampled Continuous-time ARMA Systems,” *IEEE Signal Processing Letters,* vol. 11, Febr. 2004, pp. 197–200.

[108] W. Lee, K. Cheun, and S. Choi, “A Hardware-Efficient Phase/Gain Tracking Loop for the Grand Alliance VSB HDTV Television,” *IEEE Trans. on Consumer Electronics*, vol. 42, no. 3, August 1996.

[109] T. Lee and A. Hajimiri, “Oscillator Phase Noise: A Tutorial,” *IEEE J. Solid-State Circuits,* vol. 35, no. 3., Mar. 2000, pp. 326–336.

[110] D. Lee, “Analysis of Jitter in Phase-Locked Loops,” *IEEE Trans. Circuits Syst. II,* vol. 49, no. 11, Nov. 2002, pp. 704–711.

[111] J. J. Lee and G. H. Freeman, “Accuracy of the Estimator of Gaussian Autoregressive Process,” *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers,* vol. 2, 3–6 Nov. 2002, pp. 1762–1766.

[112] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[113] S.-C. Liu, J. Kramer, T. Delbrück, and R. Douglas, *Analog VLSI: Circuits and Principles,* Bradford Book, 2002.

[114] H. Liu and J. McNeill, “Jitter in Oscillators with 1/f Noise Sources,” *Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS ’04),* vol. 1, pp. 773–776, May 2004

[115] J. Liu and M. West, “Combined Parameter and State Estimation in Simulation-Based Filtering,” *In Sequential Monte Carlo Methods in Practice,* A. Doucet, J. F. G. de Freitas, and N. J. Gordon, eds., New York, Springer-Verlag, 2001.

[116] H.-A. Loeliger, “Analog Decoding and Beyond,” *Proc. 2001 IEEE Information Theory Workshop,* Cairns, Australia, Sept. 2–7, 2001, pp. 126–127.

[117] H.-A. Loeliger, “Least Squares and Kalman Filtering on Forney Graphs,” in *Codes, Graphs, and Systems,* Kluwer, R. E. Blahut and R. Koetter eds., 2001.

[118] H.-A. Loeliger, “Some Remarks on Factor Graphs”, *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, 1–5 Sept., 2003, pp. 111–115.

[119] H.-A. Loeliger, “An Introduction to Factor Graphs,” *IEEE Signal Processing Magazine*, Jan. 2004, pp. 28–41.

[120] H.-A. Loeliger, *Signal and Information Processing,* Lecture Notes, Winter Semester 2004–2005, ETH Zurich.

[121] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy, "Probability Propagation and Decoding in Analog VLSI," *IEEE Trans. Information Theory,* vol. 47, pp. 837–843, Feb. 2001.

[122] V. Lottici and M. Luise, "Carrier Phase Recovery for Turbo-Coded Linear Modulations", *International Conference on Communications 2002*, New York, NJ, 28 April–2 May, 2002, vol. 3, pp. 1541–1545.

[123] F. Lustenberger, *On the Design of Analog VLSI Iterative Decoders.* PhD Thesis at ETH Zurich, Diss. ETH No 13879, Nov. 2000.

[124] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of Constructions of Irregular Gallager Codes," *IEEE Trans. on Comm.*, vol. COMM–47, no. 10, pp. 1449–1454, 1999.

[125] G. Matz and P. Duhamel, "Information-Geometric Formulation and Interpretation of Accelerated Blahut-Arimoto-type Algorithms," *Proc. 2004 IEEE Information Theory Workshop,* San Antonio, TX, USA, Oct. 24–29, 2004.

[126] M. Maxwell and M. Woodroofe, "Central Limit Theorems for Additive Functionals of Markov Chains", *The Annals of Probability*, vol. 28, no. 2, pp. 713–724, 2000.

[127] R. J. McEliece and M. Yildirim, "Belief Propagation of Partially Ordered Sets," in *Mathematical Systems Theory in Biology, Communication, Computation, and Finance, IMA Volumes in Math. & Appl.*, D. Gilliam and J. Rosenthal, Eds. Springer Verlag, 2003.

[128] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions,* Wiley, 1997.

[129] J. A. McNeill, *Jitter in Ring Oscillators*, PhD. Thesis, Boston University, 1994.

[130] C. Mead, *Analog VLSI and Neural Systems,* Addison-Wesley VLSI Systems Series, 1989.

[131] A. Mehrotra, *Simulation and Modelling Techniques for Noise in Radio Frequency Integrated Circuits*, PhD. Thesis, University of California at Berkeley, 1999.

[132] A. Mehrotra, "Noise Analysis of Phase-Locked Loops," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 49, no. 9, pp. 1309–1316, Sept. 2002.

[133] D. Megnet, H. Mathis, P. Flammant, and A. Thiel, "C/A-Code Synchronization Using Analog Feedback Shift Registers (AFSR)," Proc. ION GNSS 2004, Long Beach, CA, September 21–24, 2004, pp. 32–42.

[134] P. Merkli, *Message-Passing Algorithms and Analog Electronic Circuits.* PhD. Thesis at ETH Zurich, Diss. ETH No. 15942, April 2005.

[135] M. Meyr and S. A. Fechtel, *Synchronization in Digital Communications, Volume 1: Phase-, Frequency-Locked-Loops, and Amplitude Control,* New York, John Wiley & Sons, 1990.

[136] M. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation and Signal Processing,* New York, John Wiley & Sons, 1998, pp. 246–249.

[137] B. Mielczarek and A. Svensson, "Phase Offset Estimation Using Enhanced Turbo Decoders," *IEEE International Conference on Communications*, ICC 2002, vol. 3, 28 April–2 May, 2002, pp. 1536–1540.

[138] M. Moeneclaey, "On the True and the Modified Cramér-Rao Bounds for the Estimation of a Scalar Parameter in the Presence of Nuisance Parameters", *IEEE Transactions on Communications,* vol. 46, Nov. 1998, pp. 1536–1544.

[139] A. F. Molisch, *Wideband Wireless Digital Communications.* Prentice Hall, 2001.

[140] S. Moser, *Duality-Based Bounds on Channel Capacity,* ETH Dissertation no. 15769, ETH Zurich, Switzerland, Jan. 2005.

[141] C. R. Nassar and M. R. Soleymani, "Joint Sequence Detection and Phase Estimation Using the EM algorithm," *Conference Proceedings Canadian Conference onElectrical and Computer Engineering,* vol. 1, Sept. 1994, pp. 296–299.

[142] B. Ninness, "The Asymptotic CRLB for the Spectrum of ARMA Processes," *IEEE Transactions on Signal Processing,* vol. 51, no. 6, June 2003, pp. 1520–1531.

[143] M. J. Nissilä, S. Pasupathy, and A. Mämmelä, "An EM Approach to Carrier Phase Recovery in AWGN Channel," *IEEE International Conference on Communications (ICC 2001),* no. 1, June 2001 pp. 2199–2203.

[144] N. Noels, C. Herzet, A. Dejonghe, V. Lottici, and L. Vandendorpe, "Turbo Synchronization : an EM Algorithm Interpretation," *International Conference on Communications 2003*, Anchorage, Alaska, May 11–15, 2003, pp. 2933–2937.

[145] N. Noels, H. Steendam, and M. Moeneclaey, "The Cramér-Rao Bound for Phase Estimation from Coded Linearly Modulated Signal," *IEEE Communication Letters,* vol. 7, No. 5, pp. 207–209, May 2003.

[146] N. Noels, H. Wymeersch, H. Steendam, and M. Moeneclaey, "The True Cramér-Rao Bound for Timing Recovery from a Bandlimited Linearly Modulated Waveform with Unknown Carrier Phase and Frequency," *IEEE Transactions on Communications,* vol. 52, pp. 473–483, March 2004.

[147] N. Noels, H. Steendam, and M. Moeneclaey, "The True Cramér-Rao Bound for Carrier Frequency Estimation from a PSK Signal," *IEEE Transactions on Communications*, vol. 52, pp. 834–844, May 2004.

[148] N. Noels, H. Steendam, and M. Moeneclaey, “On the Cramér-Rao Lower Bound and the Performance of Synchronizers for (Turbo) Encoded Systems,” *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications*, Lisboa, Portugal, July 11–14, 2004.

[149] N. Noels, H. Steendam, and M. Moeneclaey, “Carrier and Clock Recovery in (Turbo) Coded Systems: Cramér-Rao Bound and Synchronizer Performance,” *EURASIP Journal on Applied Signal Processing, JASP, Special issue on Turbo Processing,* vol. 2005, pp. 972–980, May 2005.

[150] N. Noels, H. Steendam, and M. Moeneclaey, “Carrier Phase and Frequency Estimation for Pilot-Assisted Transmission: Bounds and Algorithms,” accepted for publication in *IEEE Transactions on Signal Processing*, 2005.

[151] N. Noels, H. Steendam, M. Moeneclaey, and H. Bruneel, “A Maximum-Likelihood Based Feedback Carrier Synchronizer for Turbo-Coded Systems,” *Proc. 61st IEEE Vehicular Technology Conference, VTC spring ’05,* Paper A4-3, May 29–June 1, 2005.

[152] R. Nuriyev and A. Anastasopoulos, “Analysis of Joint Iterative Decoding and Phase Estimation for the Non-Coherent AWGN Channel, Using Density Evolution,” *Proc. 2002 IEEE Information Theory Workshop,* Lausanne, Switzerland, June 30–July 5, 2002, p. 168.

[153] R. Nuriyev and A. Anastasopoulos, “Pilot-Symbol-Assisted Coded Transmission Over the Block-Noncoherent AWGN Channel,” *IEEE Trans. Comm.* vol. 51, no. 6, pp. 953–963, June 2003.

[154] H. Nyquist, “Thermal Agitation of Electric Charge in Conductors”, Phys. Rev. 32, 110 (1928).

[155] M. Okumura and H. Tanimoto, “A Time-Domain Method for Numerical Noise Analysis of Oscillators”, Proc. of the ASP-DAC, Jan. 1997.

[156] H. Park, S. I. Amari, and K. Fukumizu, “Adaptive Natural Gradient Gradient Learning Algorithms for Various Stochastic Models,” *Neural Networks,* vol. 13, 2000, pp. 755–764.

[157] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann, 1988.

[158] L. M. Pecora and T. L. Carroll, “Synchronization in Chaotic Systems,” *Physical Review Letters,* vol. 64, pp. 821–824, Feb. 1990.

[159] H. Permuter and J. M. Francos, “Estimating the Orientation of Planar Surfaces: Algorithms and Bounds,” *IEEE Trans. Information Theory,* vol. 46, Aug. 2000, pp. 1908–1920.

[160] H. D. Pfister, J. B. Soriaga, and P. H. Siegel, “On the Achievable Information Rates of Finite-State ISI Channels,” *Proc. 2001 IEEE Globecom,* San Antonio, TX, pp. 2992–2996, Nov. 25–29, 2001.

[161] D. Piccinin, P. Boffi, A. Cabas, and M. Martinelli,"Free-Space System for Metropolitan Optical Network Transparent Link," *Proc. SPIE,* vol. 4635, pp. 50–56, 2002.

[162] B. Porat and B. Friedlander, "On the Estimation of Variance for Autoregressive and Moving Average Processes," *IEEE Trans. Information Theory,* vol. 32, Jan. 1986, pp. 120–125.

[163] B. Porat and B. Friedlander, "Computation of the Exact Information Matrix of Gaussian Time Series with Stationary Random Components," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 34, no. 1, Febr. 1986, pp. 118–130.

[164] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing,* Cambridge University Press, 2002.

[165] J. G. Proakis, *Digital Communications*, 3rd ed. McGraw–Hill, 1995.

[166] M. H. Protter and C. B. Morrey, Jr., *Intermediate Calculus,* Springer Verlag New York, 1985.

[167] B. L. S. Rao, "Cramér-Rao-Type Integral Inequalities for General Loss Functions", Test, Sociedad de Estadística e Investigación Operativa, vol. 10, no. 1, pp. 105–120, 2001.

[168] I. Rapoport and Y. Oshman, "Recursive Weiss-Weinstein Lower Bounds for Discrete-Time Nonlinear Filtering," *43d IEEE Conference on Decision and Control,* Atlantis, Paradise Island, Bahamas, pp. 2662–2667, Dec. 14–17, 2004.

[169] S. Reece and D. Nicholson, "Tighter Alternatives to the Cramér-Rao Lower Bound for Discrete-Time Filtering," *Eighth IEEE International Conference on Information Fusion,* Philadelphia, Penn., 2005.

[170] I. Reuven and H. Messer, "A Barankin-type lower bound on the estimation error of a hybrid parameter vector," *IEEE Trans. Information Theory,* vol. 43, no. 3, pp. 1084–1093, 1997.

[171] C. Robert and G. Casella, *Monte Carlo Statistical Methods,* Springer Texts in Statistics, 2nd ed., 2004.

[172] Y. Rockah and P. Schultheiss, "Array Shape Calibration Using Sources in Unknown Locations. I. Far-field sources," *IEEE Trans. Acoust., Speech, and Signal Proc.,* vol. 35, no. 3, pp. 286–299, 1987.

[173] P. Rusmevichientong and B. Van Roy, "An Analysis of Belief Propagation on the Turbo Decoding Graph with Gaussian Densities," *IEEE Trans. Information Theory*, vol. IT–47, no. 2, pp. 745–765, 2001.

[174] W. Sandham and M. Leggett (editors), *Geophysical Applications of Artificial Neural Networks and Fuzzy Logic (Modern Approaches in Geophysics)*, Kluwer Academic Publishers, 2003.

[175] R. Sarpeshkar, T. Delbrück, and C.A. Mead. "White Noise in MOS Transistors and Resistors," *IEEE Circuits and Devices,* Nov. 1993, pp. 23–29.

[176] M. J. Schervish, *Theory of Statistics,* Springer Verlag, 1995.

[177] B. Schölkopf and A. J. Smola, *Learning with Kernels,* MIT Press, 2002.

[178] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, July/Oct. 1948.

[179] V. Sharma and S. K. Singh, "Entropy and Channel Capacity in the Regenerative Setup with Applications to Markov Channels", *Proc. 2001 IEEE Int. Symp. Information Theory,* Washington, DC, USA, June 24–29, 2001, p. 283.

[180] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis,* Cambridge University Press, 2004.

[181] O. Shental, N. Shental, and S. Shamai (Shitz), "On the Achievable Information Rates of Finite-State Input Two-Dimensional Channels with Memory," *Proc. 2005 IEEE International Symposium on Information Theory,* to appear.

[182] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications,* vol. 3, Rockville, Maryland, USA, Computer Science Press, 1985.

[183] H. Steendam and M. Moeneclaey, "Low-SNR Limit of the Cramér-Rao Bound for Estimating the Time Delay of a PSK, QAM or PAM Waveform," *IEEE Communications Letters,* vol. 5, pp. 31–33, Jan. 2001.

[184] H. Steendam and M. Moeneclaey,"Low-SNR Limit of the Cramér-Rao Bound for Estimating the Carrier Phase and Frequency of a PAM, PSK or QAM Waveform," *IEEE Communications Letters,* vol. 5, pp. 218–220, May 2001.

[185] H. Steendam, N. Noels, and M. Moeneclaey, "Iterative Carrier Phase Synchronization for Low-Density Parity-Check Coded Systems", *International Conference on Communications 2003*, Anchorage, Alaska, May 11–15, 2003, pp. 3120–3124.

[186] P. Stoica and R. Moses, "On Biased Estimators and the Unbiased Cramér-Rao Lower Bound," *IEEE Trans. Signal Proc.*, vol. 21, pp. 349–350, Oct. 1990.

[187] P. Stoica and Y. Selen, "Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: a refresher," *IEEE Signal Processing Mag.,* pp. 112–114, Jan. 2004.

[188] M. Strevens, *Bigger than Chaos: Understanding Complexity through Probability,* Harvard University Press, 2003.

[189] I. Sutskover, S. Shamai, and J. Ziv, “A Novel Approach to Iterative Joint Decoding and Phase Estimation,” *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, Brest, France, 1–5 Sept., 2003, pp. 83–86.

[190] R. M. Tanner, “A Recursive Approach to Low-Complexity Codes,” *IEEE Trans. on Inform. Theory*, vol. IT–27, pp. 533–547, Sept. 1981.

[191] R. M. Taylor Jr., B. P. Flanagan, and J. A. Uber, “Computing the Recursive Posterior Cramér-Rao Bound for a Non-Linear Non-Stationary System”, *Proc. of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 6, April 6–10, 2003, pp. 673–676.

[192] P. Tichavský, C. H. Muravchik, and A. Nehorai, “Posterior Cramér-Rao Bounds for Discrete-Time Non-Linear Filtering,” *IEEE Transactions on Signal Processing,* vol. 46, May 1998, pp. 1386–1396.

[193] M. Tipping, “Bayesian Inference: An Introduction to Principles and Practice in Machine Learning,” In O. Bousquet, U. von Luxburg, and G. Rätsch (Eds.), *Advanced Lectures on Machine Learning,* pp. 41–62. Springer, 2004.

[194] D. M. Titterington, “Recursive Parameter Estimation Using Incomplete Data,” *R. Roy. Stat. Soc.,* vol. 46(B), pp. 256–267, 1984.

[195] P. D. Tuan, “Cramér-Rao Bounds for AR Parameter and Reflection Coefficients Estimators,” *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 37, no. 5, May 1989, pp. 769–772.

[196] M. Unser, A. Aldroubi, and M. Eden, “B-spline Signal Processing. I. Theory”, *IEEE Transactions on Signal Processing,* vol. 41, no. 2, pp. 821–833, Feb. 1993.

[197] G. Ungerboeck, “New Application for the Viterbi Algorithm: Carrier Phase Tracking in Synchronous Data Transmission Systems,” *Proc. Nat. Telecomm. Conf.,* pp. 734–738, 1974.

[198] P. Vanassche, G. Gielen, and W. Sansen, “On the Difference Between Two Widely Publicized Methods for Analyzing Oscillator Phase Behavior,” *Proceedings of the 2002 IEEE/ACM international Conference on Computer-aided Design,* San Jose, California, pp. 229–233, 2002.

[199] H. L. Van Trees, *Detection, Estimation, and Modulation Theory: Part I,* Wiley, New York, 1968.

[200] B. Vigoda, *Analog Logic: Continuous-Time Analog Circuits for Statistical Signal Processing.* PhD. Dissertation. Cambridge, MA: Massachusetts Institute of Technology. September, 2003.

[201] B. Vigoda, J. Dauwels, M. Frey, N. Gershenfeld, T. Koch, H.-A. Loeliger, P. Merkli, *Synchronization of Pseudo-Random Signals by Forward-Only Message Passing with Application to Electronic Circuits, IEEE Trans. Information Theory,* to appear.

[202] A. Viterbi, *Principles of Coherent Communication*, McGraw-Hill, 1966.

[203] P.O. Vontobel, *Algebraic Coding for Iterative Decoding,* PhD. Thesis at ETH Zurich, Diss. ETH No. 14961, ETH Zurich, 2003.

[204] P. O. Vontobel, A. Kavčić, D. Arnold, and H.-A. Loeliger, "Capacity of Finite-State Machine Channels," submitted to *IEEE Trans. Inform. Theory*, Nov. 2004.

[205] M. J. Wainwright, T. Jaakkola, and A. S. Willsky, "A New Class of Upper Bounds on the Log Partition Function," Accepted for publication in *IEEE Trans. Information Theory*, March 2005.

[206] X. Wautelet, C. Herzet, and L. Vandendorpe, "Cramér-Rao Bounds for Channel Estimation with Symbol a Priori Information," *SPAWC'05, IEEE Workshop on Signal Processing Advances in Wireless Communications,* June 5–8, 2005, New-York, USA.

[207] E. Weinstein, M. Feder, and A. V. Oppenheim, "Sequential Algorithms for Parameter Estimation Based on the Kullback-Leibler Information Measure," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 38, Sept. 1990, pp. 1652–1654.

[208] E. Weinstein and A. J. Weiss, "A General Class of Lower Bounds in Parameter Estimation," *IEEE Trans. Information Theory,* vol. 34, pp. 338–342, 1988.

[209] Y. Weiss and W. T. Freeman, "On the Optimality of the Max-Product Belief Propagation Algorithm in Arbitrary Graphs," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 736–744, 2001.

[210] N. Wiberg, *Codes and Decoding on General Graphs.* Linköping Studies in Science and Technology, Ph.D. Thesis No. 440, Univ. Linköping, Sweden, 1996.

[211] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and Iterative Decoding on General Graphs," *Europ. Trans. Telecomm.,* vol. 6, pp. 513–525, Sept/Oct. 1995.

[212] W. Wiegerinck and T. Heskes, "Fractional Belief Propagation," *Proc. Neural Information Processing Systems (NIPS),* vol. 15, pp. 438–445, 2002.

[213] G. Winkler, *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods,* Springer, 2003.

[214] A. P. Worthen and W. E. Stark, "Unified Design of Iterative Receivers Using Factor Graphs," *IEEE Trans. Information Theory,* vol. 47, Feb. 2001, pp. 843–849.

[215] M. Wu, J. Yang, and C. Lee, "A Constant Power Consumption CMOS LC Oscillator Using Improved High-Q Active Inductor with Wide Tuning-Range," *The 47th Midwest Symposium on Circuits and Systems (MWS-CAS '04),* vol. 3, pp. 347–350, 25–28 July 2004.

[216] H. Wymeersch, “Software Radio Algorithms for Coded Transmission”, PhD Thesis, Ghent University, September 2005.

[217] E. P. Xing, M. I. Jordan, and S. Russell, “A Generalized Mean Field Algorithm for Variational Inference in Exponential Families,” *Uncertainty in Artificial Intelligence (UAI2003),* (eds. Meek and Kjaelff) Morgan Kaufmann Publishers, pp. 583–591, 2003.

[218] Y. Yamamoto, *Fundamentals of Noise Processes,* Lecture Notes, Autumn 2004, Stanford University. Available from `http://www.stanford.edu/~sanaka/flink/EEAP248/EEAP248.html.`

[219] L.-L. Yang and L. Hanzo, “Iterative Soft Sequential Estimation Assisted Acquisition of m-Sequences,” *Electronics Letters,* vol. 38, pp. 1550–1551, Nov. 2002.

[220] L.-L. Yang and L. Hanzo, “Acquisition of m-Sequences Using Recursive Soft Sequential Estimation,” *IEEE Trans. Comm.,* vol. 52, pp. 199–204, Feb. 2004.

[221] J. C. Ye, Y. Bresler, and P. Moulin, “Cramér-Rao Bounds for Parametric Shape Estimation,” *Proc. 2002 International Conference on Image Processing,* Rochester, NY, USA, Sept. 22–25, 2002, vol. 2, pp. 473–476.

[222] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Understanding Belief Propagation and Its Generalization”, *Exploring Artificial Intelligence in the New Millennium,* ISBN 1558608117, Chap. 8, pp. 239–236, January 2003.

[223] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms,” *IEEE Trans. Information Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[224] O. W. Yeung and K. M. Chugg, “An Iterative Algorithm and Low Complexity Hardware Architecture for Fast Acquisition of Long PN Codes in UWB systems,” to appear.

[225] A. Yuille, “CCCP Algorithms to Minimize the Bethe and Kikuchi Free Energies: Convergent Alternatives to Belief Propagation,” *Neural Computation,* vol. 14, pp 1691–1722, 2002.

[226] H. Zamiri-Jafarian and S. Pasupathy, “EM-Based Recursive Estimation of Channel Parameters,” *IEEE Transactions on Communications,* vol. 47, Sept. 1999, pp. 1297–1302.

[227] J. Zhang, R. A. Kennedy, and T. D. Abhayapala, “Cramér-Rao Lower Bounds for the Time Delay Estimation of UWB Signals”, *Proc. 2004 IEEE International Conference on Communications,* Paris, France, June 20–24, 2004, vol. 6, pp. 3424–3428.

[228] A. M. Zoubir and A. Taleb, “The Cramér-Rao Bound for the Estimation of Noisy Phase Signals,” *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001 (ICASSP ’01)*, vol. 5, May 2001, pp. 3101–3104.

Index

About the Author

Justin Dauwels was born in Eeklo, Belgium, on November 2, 1977. After visiting primary school in Zelzate, Belgium, he attended the Sint-Barbaracollege in Gent with Ancient Greek and Mathematics as majors. In 1995, he joined Gent University, Belgium, to study Engineering Physics. He was an exchange student in 1999–2000 at the the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland. In Summer 2000, he carried out his diploma thesis at the Institute of Neuroinformatics (ETH Zurich/ University of Zurich) in Zurich. In September 2000, he graduated from Gent University, Belgium, and later that year, he started as research and teaching assistant at the Signal and Information Processing Laboratory (ISI) at the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland. In Fall 2003, he was a visiting scientist at the Media Laboratory at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, and in January 2004 at the Digital Communications Laboratory at Gent University, Belgium. In Spring 2004, he was an intern at the Mitsubishi Electric Research Labs, Cambridge, MA, USA. He is currently a JSPS Post-Doctoral Fellow at the RIKEN Brain Science Institute in Wako-shi, Saitama, Japan.