Faculteit Toegepaste Wetenschappen

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter : Prof. Dr. ir. Herwig Bruneel

Algoritmes voor Software Radio, geschikt voor gecodeerde transmissie

Software Radio Algorithms for Coded Transmission

door

ir. Henk Wymeersch

Promotoren: Prof. Dr. ir. Marc Moeneclaey en Prof. Dr. ir. Heidi Steendam

PROEFSCHRIFT INGEDIEND BIJ DE FACULTEIT TOEGEPASTE WETENSCHAPPEN VAN DE UNIVERSITEIT GENT TOT HET BEHALEN VAN DE ACADEMISCHE GRAAD VAN DOCTOR IN DE TOEGEPASTE WETENSCHAPPEN

Academiejaar 2004-2005

Acknowledgments

This manuscript is the apogee of just under four years spent at the TELIN department. During this time, I was fortunate enough to have two excellent advisors: Marc Moeneclaey (you the man!) and Heidi Steendam. I owe, more than I can say, to their continuing guidance, support and insights. Thank U.

I'd also like to thank my colleagues, nay, friends, at the DIGCOM research group: Mamoun Guenach (Mr CDMA and SAGE enthusiast), Frederik Simoens (my partner in crime), Nele Noels (Computer of the Holy CRB) and Frederik Vanhaverbeke (maker of desserts extra-ordinaire, and proof-reader of this thesis). Thank U.

Kudos to my circle of friends, mainly for putting up with me all this time. It can't have been easy. A special mention for Diter Wuytens (proof-reader of many chapters, quod non). Thank U.

Most importantly, my family: without them, I would not be where I am today; I simply wouldn't be. I love you all.

And finally, everyone who was directly or indirectly involved with my research, and whom I've forgotten to thank explicitely: you know who you are and I thank you.

Henk Wymeersch - May 9th 2005.

Contents

Notations

$p(x\,\vert y)$	notational shorthand for $p_{X\vert Y}\ (X = x\,\vert Y = y)$
$\sum_{\sim\{A\}} f(B)$	sum over $B \setminus A$
$\sum_{\mathbf{a}:a_k} f(\mathbf{a})$	sum over all $\mathbf{a}$ for which the k-th component equals a_k
$E_x\left[f(x,y,z)\,\vert z\right]$	expectation of $f(x,y,z)$ w.r.t. x, conditioned on $z: \int p(x\vert\, z)\, f(x,y,z)\, dx$
$Var_x\left[f(x,y,z)\,\vert z\right]$	variance of $f(x,y,z)$ w.r.t. x, conditioned on z
$\Im\{x\}$	the imaginary part of x
$\Re\{x\}$	the real part of x
x^*	complex conjugate of x
$\hat{x}$	an estimate of x
$\mathbf{x}$	a vector $\mathbf{x}$
x_k	the k-th element of some vector $\mathbf{x}$
$\mathbf{X}$	a matrix $\mathbf{X}$
$\mathbf{X}^T$	the transpose of $\mathbf{X}$
$\mathbf{X}^H$	the transpose complex conjugate of $\mathbf{X}$: $\mathbf{X}^H = (\mathbf{X}^*)^T$
$\mathbf{X}_{n,m}$	element at row n, column m of $\mathbf{X}$
$\Vert\mathbf{x}\Vert$	norm of $\mathbf{x}$: $\Vert\mathbf{x}\Vert = \sqrt{\sum_k \vert x_k\vert^2}$
$X(f)$	the Fourier transform of the signal $x(t)$
$X\left(e^{j2\pi f T_s}\right)$	the discrete Fourier transform of the sample sequence $\{x(kT_s)\}$
$x \doteq y$	x is a notational shorthand for y (or vice versa)
$x \propto y$	x is proportional to y: x is equal to y up to irrelevant additive and multiplicative terms
$x \sim \mathcal{N}\left(\mu, \sigma^2\right)$	x is a real random variable drawn from a Gaussian distribution with mean μ and variance σ^2
$x \sim \mathcal{CN}\left(\mu, 2\sigma^2\right)$	x is a complex random variable, with independent real and imaginary parts, such that $\Re\{x\} \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and $\Im\{x\} \sim \mathcal{N}\left(\mu, \sigma^2\right)$
$(n,m)_8$	n and m in octal notation
$[a,b]$	a continuous interval from $a \in \mathbb{R}$ to $b \in \mathbb{R}$
$\mathbf{1}_{N\times M}$	an $N \times M$ matrix of ones
$\mathbf{0}_{N\times M}$	an $N \times M$ matrix of zeros
$\mathbf{I}_N$	an $N \times N$ identity matrix
$\arg(x)$	the phase of the complex number x
$\arg\max_x f(x)$	the value of x that maximizes $f(x)$

Glossary

A	amplitude
α	complex path gain
$CIC\left(R,L\right)$	CIC filter with decimation factor R and order L (Chapter 4 and 7)
$\mathbf{d}_e$	the parameter vector to be estimated (Chapter 8)
$\mathbf{d}_n$	the nuisance parameter (Chapter 8)
$\mathbf{d}_c$	the complete data (Chapter 8)
$\mathbf{d}_m$	the missing data (Chapter 8)
$\mathbf{d}_h$	the hidden data (Chapter 8)
$\delta\left(t\right)$	the Dirac distribution: $\int h\left(t\right)\delta\left(t-t_0\right)dt = h\left(t_0\right)$
δ_k	Dirac delta: for $k \in \mathbb{Z}$: $\delta_0 = 1$, $\delta_{k\neq 0} = 0$
E_s	energy per transmitted symbol
E_b	energy per information bit
E_c	energy per coded bit
f_c	carrier frequency
f_{IF}	intermediate frequency
$g\left(t\right)$	obtained by matched filtering $h\left(t\right)$: $g\left(t\right) = \int h^*\left(-u\right)h\left(t-u\right)du$
$h_{BB}\left(t\right)$	baseband channel impulse response
$h\left(t\right)$	overall impulse response: $h\left(t\right) = \sqrt{E_s}\int p\left(u\right)h_{BB}\left(t-u\right)du$
θ	carrier phase
$I\left[P\right]$	indicator function: $I\left[P\right] = 1$ when $P =$ true and $I\left[P\right] = 0$ otherwise
L	number of paths in multi-path channel
M	number of points in the signaling constellation Ω
$\mu_{f\to a}\left(a'\right)$	message from function node f to variable edge a, evaluated in a'
$\mu_{a\to f}\left(a'\right)$	message from variable edge a to function node f, evaluated in a'
N	oversampling factor (Chapters 5-7)
N_b	number of information bits per burst
N_s	the number of symbols per burst
N_0	AWGN power spectral density
N_c	number of coded bits per burst
Ω	the signaling constellation
$p\left(t\right)$	the transmit pulse
$p_I\left(t\right)$	the interpolation pulse
$q\left(t\right)$	obtained by matched filtering $p\left(t\right)$:$q\left(t\right) = \int p^*\left(-u\right)p\left(t-u\right)du$
$r_X\left(t\right)$	received signal, with X the baseband, IF or bandpass representation
$s_X\left(t\right)$	transmitted signal with X the baseband, IF or bandpass representation
σ^2	noise variance per real dimension, often $\sigma^2 = N_0/\left(2E_s\right)$
t	time
T	the symbol duration ($1/T$ is the symbol rate)
$1/T_s$	the sampling rate
τ	propagation delay
$y\left(t\right)$	obtained by filtering $r\left(t\right)$ with $p^*\left(-t\right)$: $y\left(t\right) = \int p^*\left(-u\right)r\left(t-u\right)du$
	Sometimes: obtained by filtering $r\left(t\right)$ with $h^*\left(-t\right)$: $y\left(t\right) = \int h^*\left(-u\right)r\left(t-u\right)du$

Abbreviations and acronyms

AAF	Anti-Aliasing Filter
ADC	Analog to Digital Conversion (or Converter)
APP	A Posteriori Probabilities
ASP	Analog Signal Processing
AWGN	Additive White Gaussian Noise
BB	BaseBand
BER	Bit Error Rate
BP	BandPass
CDMA	Code Division Multiple Access
CIC	Cascaded Integrator Comb
DAC	Digital to Analog Conversion (or Converter)
DTFT	Discrete-Time Fourier Transform
DSP	Digital Signal Processing
DS/SS	Direct-Sequence/Spread-Spectrum
EM	Expectation Maximization (as in the EM algorithm)
FER	Frame Error Rate
FT	Fourier Transform
IF	Intermediary Frequency
iid	Independent and identically distributed
IP	InterPolator
MF	Matched Filter
MSEE	Mean Squared Estimation Error
MMSE	Minimum Mean Squared Error
PSD	Power Spectral Density
PSK	Phase Shift Keying (e.g, M-PSK, BPSK, QPSK)
RF	Radio Frequency
SAGE	Space Alternating Generalized EM algorithm
SP	Sum-Product (as in the SP algorithm)
SR	Software Radio
SRC	Sample Rate Conversion

Part I

Introduction

Chapter 1

Introduction

1.1 An evolution in communications

In the last decade, the way we communicate, access and distribute information has changed dramatically. The Internet is now virtually ubiquitous. Cell-phones, digital cameras, laptops, GPS and portable media players are becoming commonplace, and are constantly transforming, changing, merging. There is a powerful economic drive to recombine existing techniques and standards, to create wireless devices that people can use for a variety of tasks (an idea aptly referred to as 'convergence'). One of the goals is a device that is able to support different standards (such as GSM, DECT, CDMA, UMTS, WiFi), is upgradable to future standards, can support heterogeneous services (e.g., narrow-band voice and broadband data services, entertainment, GPS, etc.), at low cost. Furthermore, this device should be able to operate under the most adverse conditions, with very low power consumption, using the least possible bandwidth, and so forth [1,2]. An important aspect is that of *adaptivity*: how, and according to which criteria can the transmission-parameters (e.g., transmit power, data rate, etc.) be changed over time in an efficient manner? Many current standards strive to meet at least a subset of the abovementioned requirements.

In the meantime, the communications research community is developing new ideas at an ever increasing pace. Although the bulk of this research will most likely become nothing more than historical record, every once and a while a truly remarkable idea shakes up the community, creating entire new fields of research and connecting existing ones. In the field of digital communications, it all started around the time (for a historical overview, see [3]) of Claude Shannon's 1948 landmark paper: 'A Mathematical Theory of Communication' [4]. Not only did Shannon singlehandedly create the field of Information Theory, he also changed the way people thought about communication. Rather than considering the transmission of electromagnetic waves, Shannon envisaged digital information. The concept of sending information in digital form (e.g., as bits) was formalized at that time. Shannon also derived a bound, now known as the Shannon Bound, relating a channel to the information rate; the Shannon Bound basically states how many bits per second one can reliably transmit over a given channel. This bound is still used as a benchmark for state-of-the-art communication systems [5]. By defining a new communication paradigm, Shannon helped in laying the foundation for a large set of related problems, many of which are still relevant today. For instance, the field of coding theory, concerned with how we can protect the bits that are to be transmitted over the channel. The field of multi-user communication, where one allows multiple users to share the same channel. And of course other topics, too many to list here.

Probably the most groundbreaking contribution in the late 20th century is the concept of iterative processing (*turbo-processing*). Although this idea has been latently present since the early 1960s [6], it took until 1993 [7] before its potential was fully recognized, and before the processing power was available to implement the corresponding algorithms. The idea is conceptually simple: a complex task is divided in simple sub-tasks. To solve the complex task, one solves the sub-tasks, exploiting information available from the solutions of other sub-tasks. Although deceptively straightforward, the reader should appreciate the enormous impact the turbo-concept has had in the last decade. For instance, the idea was originally applied to the problem of channel coding, resulting in the development of a novel type of code (a turbo-code, naturally). For the first time since Shannon's 1948 paper, a research group was able to create a practically implementable code, that could approach the Shannon Bound[1]! Later, the turbo-principle was applied in virtually every area of digital communications, leading to techniques such as turbo-synchronization, turbo-multi-user-detection, turbo-equalization, etc. The turbo-principle was originally developed in a rather adhoc way. Only recently a mathematical framework was proposed confirming, up to a point, the validity of these algorithms. This framework goes under the name of 'factor graphs' [8,9], and provides insightful ways to develop

[1]At the International Communications Conference in 1993, where Claude Berrou et al. first presented their turbo-codes, the idea was met with significant scepticism. Only when the results had been independently verified, was the turbo-concept fully accepted.

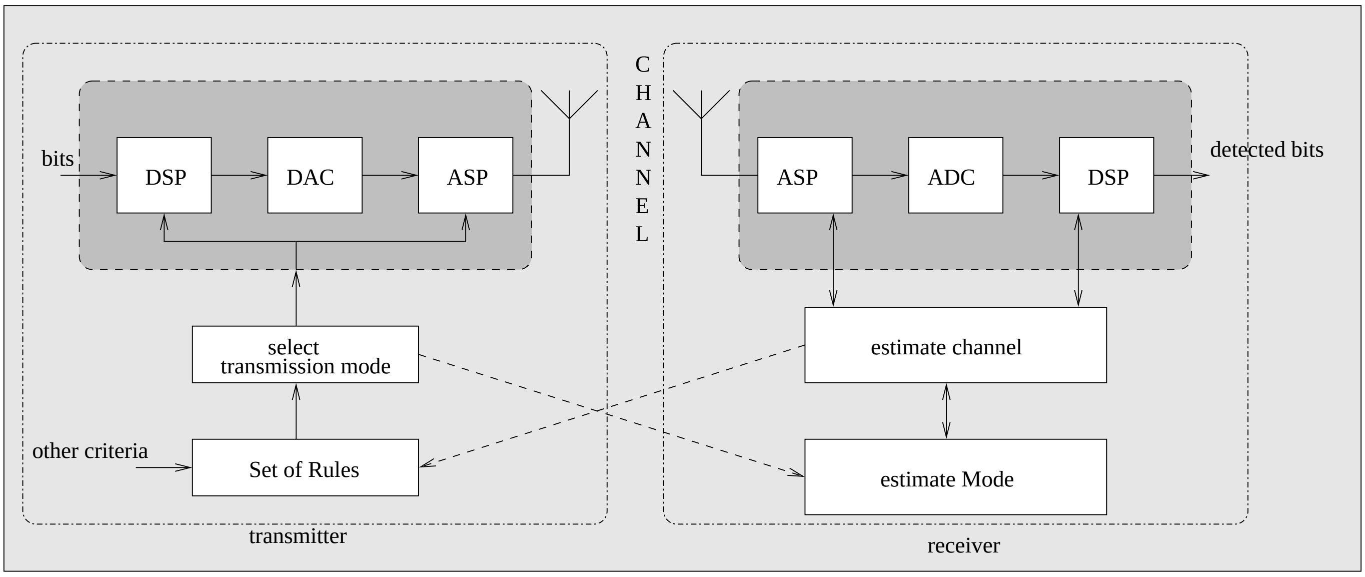


Figure 1.1: *A generic Multi-Mode Transceiver*

iterative algorithms, based on a graphical representation of a problem or system.

Let us take a step back. On the one hand there is a need from the industry to create powerful, adaptive devices (requiring powerful, adaptive algorithms). On the other hand, a plethora of new iterative algorithms and techniques from the communications research community are available. In this dissertation, we will forge some links between iterative processing and adaptive transceivers.

In the remainder of this chapter, we will give a brief description of the transmission schemes under consideration, and place them in their proper context. A number of practical problems related to such receivers will be touched upon. We end with an overview of the organization of this dissertation.

1.2 Multi-mode transmission and Software Radio

1.2.1 Concept

In adaptive transmission, the transmitter can decide, based on a set of rules, to change the way digital information is transmitted. For instance, the transmitter could increase the transmit power depending on the channel conditions, to ensure a target quality of service (QoS). The transmitter could also adapt the data rate, depending on the type of content that needs to be transferred [10]. The different regimes under which a transmitter can operate will be named the *transmit modes*. Determining a set of rules according to which modes should be selected is referred to as *link adaptation*, and deals mainly with information-theoretic aspects of multi-mode transmission. The process of modifying (through filters or software) the transmitter and receiver to operate in a given mode will be referred to as *mode adaptation*.

A generic multi-mode transceiver is shown in Fig. 1.1. The transmitter selects a transmit mode, according to the link adaptation strategy. For instance, the transmitter could decide to adapt the transmit power and modulation based on certain channel state information, in order to reduce the overall transmit power and thus increase battery life. The transmitter can obtain the channel state from the receiver through some separate channel (shown in dashed in Fig. 1.1). Once a transmit mode has been selected, a sequence of bits is converted to a signal corresponding to that mode. The signal is generated through a combination of Digital and Analog Signal Processing (DSP and ASP, respectively), separated by a Digital to Analog Converter (DAC). Both analog and digital components may depend on the transmit mode. Then, the signal is transmitted to the receiver over a channel, in this example using an antenna[2]. The signal propagates through the channel and arrives at the receiver, corrupted by thermal noise. The main goal of the receiver is detecting the bits. Often the most efficient way to do this, is first estimating the channel state (and possibly the transmit mode), and only then detecting the bits, assuming the estimates to be correct. In some scenarios, the transmitter can forward the transmit mode to the receiver through some separate channel (again

[2]Or some other means, such as optical fiber, coax cable, telephone line etc.

shown in dashed in Fig. 1.1). Similar to the transmitter, the receiver also has analog and digital components, now separated by an Analog to Digital Converter (ADC).

1.2.2 Practical issues

As with many ideas, the concept of adaptive transmission is not new. At the time of its conception (some 40 years ago), there was only very limited interest [11, 12]. Only recently, with the availability of suitably powerful hardware, has adaptive transmission been put back on the agenda (see [13] and references therein).

Despite the presence of powerful hardware, mode adaptation at the transmitter and the receiver should be performed in an efficient manner. It would not do to design transceivers with different analog and digital components for each possible mode - the cost would simply be prohibitive. In fact, mode adaptation should be performed, as much as possible, in the digital rather than the analog part of the receiver: analog components cannot easily be adapted, cannot benefit from Moore's law and are generally less reliable and less cost-effective than their digital counterparts.

Shifting tasks to the digital domain goes hand in hand with the concept of Software Radio (SR) [14]. Originally conceived for military applications, the idea of Software Radio is to digitize signals close to the antenna and let all processing be performed by digital signal processors, programmable by software, operating on a suitable hardware platform. Adapting the radio to operate in a new mode (or even a new standard), simply requires loading different software. Some critical functionalities of Software Radios are [15]:

- *Analog to digital and digital to analog conversion:* when the ADC is moved closer to the antenna, the digitization bandwidth needs to be increased, as well as the dynamic range. Additionally, power consumption of ADC and DAC components may be an issue. Problems related to ADC and DAC are outside of the current scope.
- *Sample rate conversion:* different parts of the receiver may operate at different rates. The process of converting samples at one rate to samples at another rate is known as Sample Rate Conversion (SRC).
- *Channelization:* isolating independent communication channels contained within a signal. Out-of-band as well as in-band interferers should be suppressed, either in the digital or in the analog domain.

It is expected that future multi-mode receivers will be implemented as Software Radios. For that reason, we will not distinguish between these two types of receivers.

From the exposition above, it has become clear that the task of receiver and transmitter are somewhat asymmetric: the transmitter has perfect knowledge of the data and transmit mode, but only imperfect (or no) knowledge of the channel state. The receiver, on the other hand, does not know the data, has only imperfect knowledge of the channel state and may or may not know the transmit mode. In general, the task of the receiver is much more complex than that of the transmitter.

1.3 Iterative techniques

The (multi-mode) Software Radio described above does not really require any advanced estimation or detection algorithms. Standard approaches [16,17] will do just fine: one would first estimate the transmit mode and the channel parameters. Then, the receiver is adapted to the transmit mode, so that standard operations such as equalization and timing correction can be performed. Finally, the bits are detected.

However, when we include the aspect of coding, everything changes dramatically. Powerful codes have the property of achieving very reliable transmission with very low transmit power. Very low transmit power results in very low received power. Unfortunately, conventional algorithms, which have satisfactory performance for uncoded transmission (requiring high transmit power for reliable transmission), are now unable to cope: due to the low power at the receiver, these algorithms become very unreliable. Since state-of-the-art error-correcting codes are not very robust against estimation errors, new approaches are called for.

A way around this problem is applying the turbo-principle to equalization, data detection and estimation [18, 19]. Roughly speaking, one employs a set of standard algorithms to roughly estimate, equalize and perform data detection. We then have some additional information regarding the transmitted data. This information can then be used to improve the reliability of estimation and equalization algorithms. This in turn results in more reliable information to be provided to the decoder, and so forth. Such iterative techniques have been applied since the advent of turbo-codes in the early 1990s. However, most iterative estimation algorithms were developed in an ad-hoc way, and are only useful for very specific applications. A mathematical framework was still missing. Developing such iterative estimation techniques for multi-mode transceivers is an important topic and covered in part in this dissertation.

1.4 Overview

This dissertation deals with two important tasks of the receiver. First of all, the receiver has to determine the channel state and the transmit mode. Subsequently, the receiver has to adapt to the channel state and transmit mode in an efficient manner. Although we will mainly focus on receiver algorithms, some sections dealing with receiver design may also be applied to the transmitter.

The text consists of four parts. The main contributions are in Part II and III, which can be read (more or less) independently from one another. Each part starts with a chapter dealing with some background information, useful to fully appreciate the contributions in later chapters of that same part. This way, the reader can read through the chapters without getting distracted by the mathematical details.

Part I

In the first part, we will introduce a mathematical description of the system model (Chapter 2) for the coded multi-mode transmission scheme under consideration, followed by a more detailed look into some problems that are the main motivations for this research. The first part ends with a description of factor graphs, which form a unifying framework for many iterative algorithms (Chapter 3), and which will be heavily relied upon throughout this manuscript.

Part II

The second part deals with adaptation, i.e., how can a receiver which perfectly knows the transmit mode and channel state efficiently detect the data? We start with some basic material, pertaining to analog and digital signal processing (Chapter 4) which will be useful for the remainder of Part II. This is followed by a description of how a receiver can cope with adverse channel conditions (Chapter 5). We then move on to some aspects related to mode adaptation, exposing the critical issue of rate-adaptation (Chapter 6). A separate chapter is devoted to the comparison of different multi-rate receivers in terms of performance and complexity (Chapter 7).

Part III

Part III considers the problem of iterative estimation. We start again with some basic material (Chapter 8), describing general estimation techniques. This is followed by a detailed description of so-called code-aided estimation algorithms in Chapter 9. As these algorithms tend to exhibit problems related to convergence and computational complexity, we devote Chapter 10 to show how these issues can be resolved. As a by-product, we will describe a powerful algorithm for estimating discrete parameters (Chapter 11). Part III ends with a presentation of some relevant performance results (Chapter 12).

Part IV

We end our thesis with Part IV, describing some open problems and proposals for future work (Chapter 13), before wrapping things up with our final conclusions (Chapter 14). Chapter 14 also contains a complete list of own publications in international refereed conferences and journals.

Chapter 2

System Model

2.1 Introduction

In all communication systems, we are faced with the following problem. A transmitter wishes to send (digital) information to a receiver. For this communication, they have been allocated a certain channel (e.g., a carrier frequency, a maximum bandwidth, a physical link, time slots, etc.). Within the channel constraints, they have the possibility to dynamically change the way how the data is transmitted. This can be achieved by employing multiple transmit modes. This includes changing the number of bits per second (the data rate), the way how the information is represented (coding, modulation) and the transmit power [20].

Although a multi-mode transmission scheme enables the transmitter to dynamically adapt to the changing environment, it creates some extra challenges for the receiver. On top of the conventional operations the receiver has to perform (such as channel estimation, synchronization and data detection), it now also has to determine the transmit mode and adapt itself to that mode in an efficient manner. This readily defines two tasks, not present in conventional mono-mode receivers: mode detection and mode adaptation.

As mentioned before, besides these new tasks, the receiver still has to fulfill the functions required of all mono-mode receivers. The transmitted signal is corrupted in many ways as it passes through the physical channel. The most efficient way for a receiver to recover the transmitted data, is first to estimate how the signal is corrupted and then take measures to undo this corruption [16]. In addition, the received signal has to be synchronized to the transmitted signal. This means that parameters such as propagation delays and carrier phases need to be estimated and compensated for.

Clearly, the receiver has to perform two types of tasks in order to detect the data. The first is estimation of unknown parameters (transmit modes, channel parameters and synchronization parameters). Once these parameters have been estimated, the second task of the receiver is modifying itself to compensate for these parameters (resp. mode adaptation, equalization, timing- and carrier phase correction) in order to perform data detection. We will name these tasks *estimation* and *adaptation*, respectively.

In this chapter we will describe the transmitter, the channel model and the front-end of the receiver. Relevant notations and terminology are introduced. The model we present is a fairly standard one, mainly based on [16, 21] and depicted in Fig. 2.1.

2.2 Transmitter

2.2.1 Signal generation

The transmitter sends a sequence ($\mathbf{b}$) of N_b information bits to the receiver. The bits are first encoded, yielding a code sequence $\mathbf{c}$ of length N_c, where $\mathbf{c}$ and $\mathbf{b}$ are related by the one-to-one mapping χ: $\mathbf{c} = \chi(\mathbf{b})$. The ratio N_b/N_c is referred to as the code rate. The coded bits are then mapped to a sequence of N_s complex symbols $\mathbf{a} = \varphi(\mathbf{c})$, taken from an M-point constellation[1] denoted as Ω. The symbols are shaped by a unit-energy transmit pulse $p(t)$. The resulting complex baseband signal can be written as

$$s_{BB}(t) = \sqrt{E_s} \sum_{k=0}^{N_s-1} a_k p(t - kT) \tag{2.1}$$

[1]For the moment, a constellation can be thought of as a collection of M points in the complex plane.

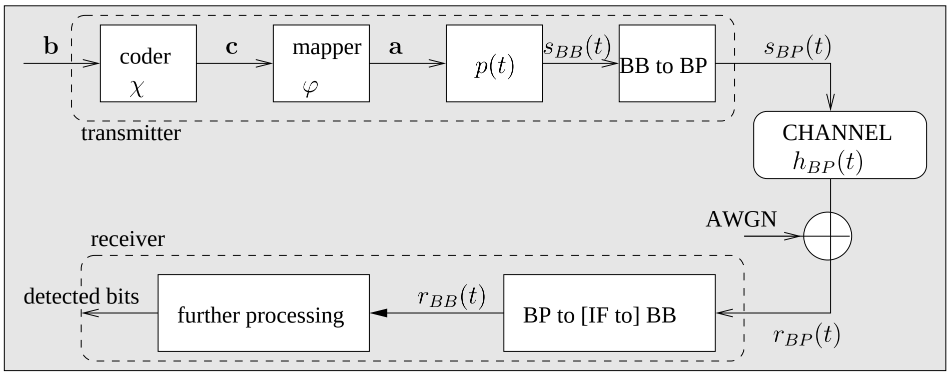


Figure 2.1: *Model of transmitter and receiver*

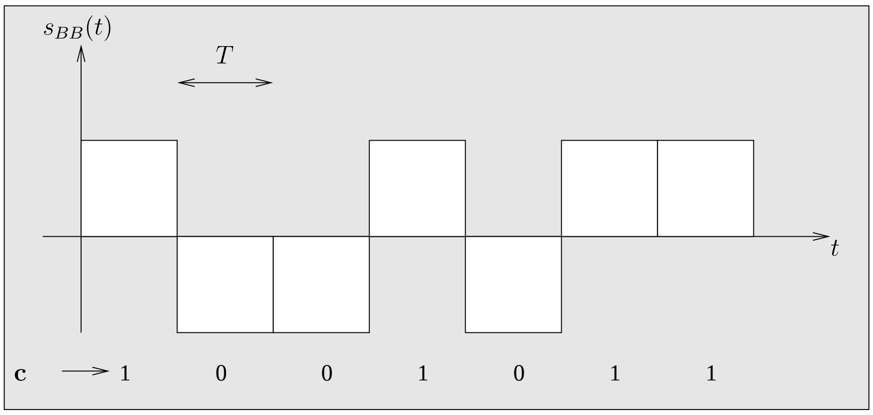


Figure 2.2: *A baseband signal with block-pulse* $p\left(t\right)$

where E_s and $1/T$ denote the transmit energy per symbol and the symbol rate, respectively. The transmit pulse has a (one-sided) bandwidth B. An example of a baseband signal with $\Omega = \{-1, +1\}$ and a block-pulse is shown in Fig. 2.2.

We distinguish between the energy per information bit (E_b), the energy per coded bit (E_c) and the energy per symbol (E_s): these are related by $E_c \log_2 M = E_s$ and $E_b N_b / N_c = E_c$.

The complex baseband signal is now converted to a real bandpass signal[2], with carrier frequency f_c. This yields:

$$s_{BP}\left(t\right) = \Re\left\{\sqrt{2} s_{BB}\left(t\right) e^{j2\pi f_c t}\right\} \tag{2.2}$$

$$= \frac{\sqrt{2}}{2}\left(s_{BB}\left(t\right) e^{j2\pi f_c t} + s_{BB}^{*}\left(t\right) e^{-j2\pi f_c t}\right). \tag{2.3}$$

2.2.2 Mono-mode and multi-mode transmission

In mono-mode transmission, the information sequence $\mathbf{b}$ uniquely defines the transmitted signal $s_{BP}\left(t\right)$. In multi-mode transmission, $s_{BP}\left(t\right)$ can depend on other parameters. These include

- the energy per information bit E_b
- the energy per transmitted symbol E_s
- the code χ
- the modulation set Ω and mapping function φ
- the symbol rate $1/T$

[2]BB = BaseBand; BP = BandPass

- the pulse shape $p(t)$

Note that these parameters are not all independent.

2.3 Receiver

The signal $s_{BP}(t)$ propagates through a channel with real channel impulse response $h_{BP}(t)$. This channel is assumed to be time-invariant, which constrains the burst length to be sufficiently short ($N_s T$ has to be much smaller than the coherence time of the channel[3]). At the receiver, the signal is further corrupted by thermal Additive White Gaussian Noise (AWGN). The signal at the input of the receiver is given by

$$\begin{aligned} r_{BP}(t) &= \int_{-\infty}^{+\infty} s_{BP}(u)\, h_{BP}(t-u)\, du + n(t) \qquad (2.4) \\ &= \int_{-\infty}^{+\infty} \Re\left\{ \sqrt{2} s_{BB}(u)\, e^{j2\pi f_c u} h_{BP}(t-u) \right\} du + n(t) \qquad (2.5) \end{aligned}$$

where $n(t)$ is a real AWGN process with power spectral density (PSD) $N_0/2$. The received signal $r_{BP}(t)$ is often referred to as the RF-signal (Radio Frequency). The RF-signal is first down-converted to baseband:

$$\begin{aligned} \tilde{r}_I(t) &= \sqrt{2} r_{BP}(t) \cos(2\pi f_c t) \qquad (2.6) \\ \tilde{r}_Q(t) &= \sqrt{2} r_{BP}(t) \sin(-2\pi f_c t) \qquad (2.7) \end{aligned}$$

where the subscripts I and Q denote the In-phase and Quadrature component of the signal. Low-pass filtering of $\tilde{r}_I(t)$ and $\tilde{r}_Q(t)$ gives rise to[4]

$$\begin{aligned} r_I(t) &= \int_{-\infty}^{+\infty} \Re\{ s_{BB}(u)\, h_{BB}(t-u) \}\, du + w_I(t) \qquad (2.8) \\ r_Q(t) &= \int_{-\infty}^{+\infty} \Im\{ s_{BB}(u)\, h_{BB}(t-u) \}\, du + w_Q(t) \qquad (2.9) \end{aligned}$$

where $w_I(t)$ and $w_Q(t)$ are independent real Gaussian noise processes; both are white within the signal bandwidth with power spectral density $N_0/2$. The filter $h_{BB}(t)$ corresponds to a down-converted version of $h_{BP}(t)$:

$$h_{BB}(t) = h_{BP}(t)\, e^{-j2\pi f_c t}. \qquad (2.10)$$

The complex baseband signal is then given by

$$\begin{aligned} r_{BB}(t) &= r_I(t) + j r_Q(t) \\ &= \int_{-\infty}^{+\infty} s_{BB}(u)\, h_{BB}(t-u)\, du + w(t) \qquad (2.11) \end{aligned}$$

Substituting (2.1) into (2.11) yields

$$r_{BB}(t) = \sum_{k=0}^{N_s-1} a_k h(t-kT) + w(t) \qquad (2.12)$$

where $h(t) = \sqrt{E_s} \int p(u)\, h_{BB}(t-u)\, du$ is obtained by convolving $\sqrt{E_s} p(t)$ and $h_{BB}(t)$.

[3]Coherence time: loosely defined as the time during which the channel is constant. The coherence time is inversely proportional to the speed of movement and the carrier frequency f_c.

[4]$\Im\{x\} = \frac{x-x^*}{2j}$,$\Re\{x\} = \frac{x+x^*}{2}$.

RF to IF conversion

Although we did not consider this explicitly in the model above, the RF signal not only consists of a useful signal component and noise, but also of signal components with other carrier frequencies belonging to other services or users. These have to be removed through filtering using a very sharp bandpass filter. This is often difficult to realize. As an additional problem, when the carrier frequency or signal bandwidth changes (which is the case in some communication standards), the filters and oscillators need to be changed. As this is impractical, the resulting receiver cannot easily be tuned. Direct conversion from RF to baseband is therefore commonly avoided [22].

In practice, the RF signal is never down-converted to baseband in a single stage. Rather, a so-called superheterodyne receiver is used, whereby the RF signal is filtered using a coarse, tunable filter that removes image components, followed by down-conversion to an Intermediate Frequency (IF). The resulting IF signal is then processed according to the methods outlined above, where f_c needs to be replaced with the IF, $f_{IF} < f_c$:

$$r_{IF}(t) = \int_{-\infty}^{+\infty} \Re\left\{\sqrt{2} s_{BP}(u) e^{j2\pi f_{IF} u} h_{IF}(t-u)\right\} du + n(t) \tag{2.13}$$

where $h_{IF}(t)$ is an IF-equivalent representation of $h_{BP}(t)$. Possibly, the receiver will have multiple IF stages, each time reducing the IF.

When the carrier frequency of the RF signal changes, tuning the filters and oscillator frequency can now easily be accomplished [22].

2.4 Channel model

A common channel model is the multi-path model, whereby the channel impulse response consists of a number of distinct paths [16]:

$$h_{BB}(t) = \sum_{l=0}^{L-1} \alpha_l \delta(t-\tau_l) \tag{2.14}$$

where α_l and τ_l are the complex gain (embedding carrier phase and attenuation) and the propagation delay of the l-th path. We order the paths as $\tau_0 < \tau_1 < \ldots < \tau_{L-1}$. We say that the l-th path is resolvable when $\tau_{l+1} - \tau_l \gg 1/B$, where B represents the bandwidth of $p(t)$. If this is not the case, both paths are to be combined into a single path, and the complex gains added.

A channel is said to be *frequency-selective* when it has at least two resolvable paths. Otherwise the channel is frequency non-selective (also known as a *frequency-flat channel*). The delay of the first path, τ_0, corresponds to the *propagation delay* of the burst through the channel. For a frequency-flat channel, the channel model (2.14) is often written as

$$h_{BB}(t) = Ae^{j\theta}\delta(t-\tau) \tag{2.15}$$

where A, θ and τ represent the channel attenuation, carrier phase and propagation delay, respectively.

Note that the channel model may further be characterized by the joint distribution of the channel gains and propagation delays. All this *a priori* information is captured in the a priori distribution, $p(\alpha_0, \tau_0, \ldots, \alpha_{L-1}, \tau_{L-1})$.

2.5 Receiver tasks

The ultimate goal of the receiver is to recover the N_b data bits $\mathbf{b}$ from the received baseband signal $r_{BB}(t)$. In order to achieve this goal, the receiver is forced to cope with some adverse conditions. For instance both the channel $h_{BB}(t)$, and the noise statistics may be unknown to the receiver. Even for a flat channel, the oscillators at the receiver will not be synchronized with those at the transmitter, so that the propagation delay and the carrier phase need to be determined before detection can take place. Finally, the receiver needs to determine the transmit mode of the current burst.

Even if all of these parameters were known, the operation of the receiver is far from trivial: the frequency selectivity needs to be taken into account (equalization), the signal needs to be reconstructed at the correct time instants (timing correction), the detector may need to be modified to operate in the current transmit mode (mode adaptation). Hence, in order to perform data detection, two tasks need be performed:

- *estimation* (of channel parameters, noise power, synchronization parameters, transmit mode);

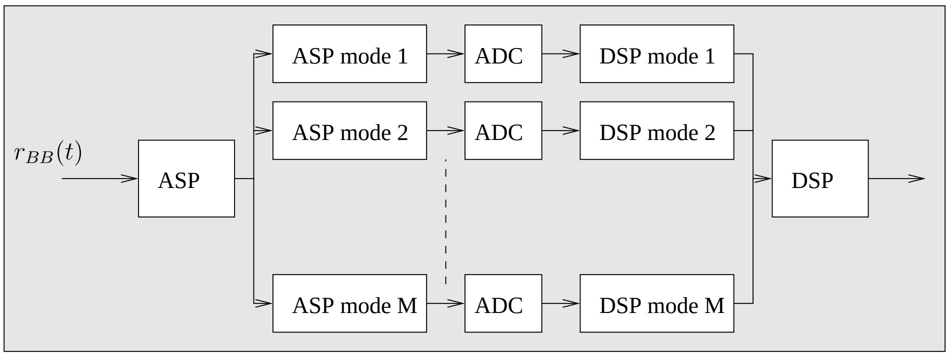


Figure 2.3: *Multi-Mode Receiver*

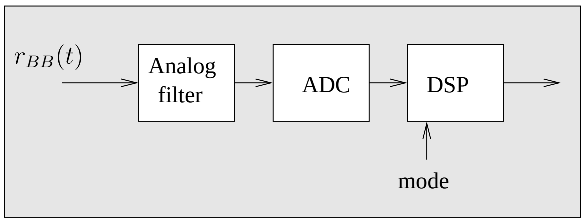


Figure 2.4: *Ideal Digital Multi-Mode Receiver*

- *adaptation* (equalization, timing correction, mode adaptation).

2.5.1 Adaptation

When the receiver knows the channel parameters, synchronization parameters, transmit mode etc, it needs to take into account these parameters to develop suitable observation models and data detection algorithms. This will be the main topic of Part II of this dissertation.

Adaptation to channel parameters and synchronization parameters deals with topics such as matched filtering, timing correction and equalization. These problems are well-known and standard solutions have been developed decades ago. The equalization problem deserves some attention: optimal techniques such as sequence estimation [23] cannot be used for many types of codes, while sub-optimal standard equalization techniques [21] are no longer sufficiently reliable in combination with state-of-the-art error-correcting codes [24]. Recently, turbo-equalization techniques have been proposed which iterate between data detection and equalization [18, 24, 25]. For the sake of completeness a short chapter (Chapter 5) is devoted to adaptation of the receiver to channel parameters and synchronization parameters.

How adaptation to the transmit mode should be performed is a problem which is fairly recent. The most straightforward way to implement such a multi-mode receiver would be as depicted in Fig. 2.3: the incoming signal is first processed in the analog domain, irrespective of the mode. Then a branch is selected, depending on the mode, so that mode-specific analog (e.g., filtering depending on the pulse-shape) and digital (e.g., equalization) signal processing can be performed. Finally, a mode-independent digital (e.g., decoding) stage may be performed. One way to significantly reduce the complexity of a multi-mode receiver is to perform no mode-dependent processing in the analog domain. This is shown in Fig. 2.4. The incoming signal would first undergo ASP, is then sampled at some rate (independent of the mode), followed by DSP. The DSP part would be implemented on some hardware platform, where a mode change corresponds to a change of the software running on that platform. For our example, the digital part would consist of loading new digital filters (depending on the pulse shape), a new equalizer and a mode-independent decoder. The whole process of mode adaptation is then transformed into efficiently changing the digital parts of the receiver through the software. As we will show, adaptation to a change in the symbol-rate may be especially critical: changing the symbol rate implies changing the transmit pulse $p(t)$. A related task is sample rate conversion (SRC): the incoming signal (either baseband or at IF) is sampled with a fixed master clock. Generally, the clock rate will be incommensurate with the symbol rate. Hence, samples taken at the master clock rate need to be converted to samples at (a multiple of) the symbol rate. This task is known as sample rate conversion, and has received much attention in the context of Software Radio [15]. We could go one step further and sample the incoming

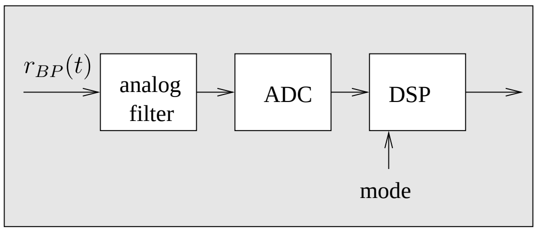


Figure 2.5: *Ideal Multi-Mode Software Radio Receiver.*

RF-signal $r_{BP}\,(t)$ directly using a wideband ADC, as depicted in Fig. 2.5. This corresponds to an ideal Software Radio, as defined in [15]. However, such an approach is not feasible with today's hardware: high resolution wideband ADCs are not yet commercially available. On the other hand, sampling the IF signal can be achieved with todays ADCs. It is well known that care needs to be taken when sampling an IF signal in order to avoid aliasing [26].

Adaptation to the transmit mode will be the main topic in Part II of this dissertation. Mode adaptation is for the most part a straightforward process, with the important exception of adaptation to the symbol rate. We pay special attention to this problem in Chapter 7, where we develop several classes of receivers which allow us to trade computational complexity for performance. These different classes of receivers are compared analytically in terms of bit-error-rate (BER) performance. This results in simple design rules for multi-rate receivers and the development of a low-complexity multi-rate receiver with a fully digital IF-sampling front-end.

2.5.2 Estimation

The problem of estimating channel parameters and synchronization parameters is present in all digital communications systems. Hence, a wide variety of algorithms have been developed over the last 50 years to perform this task. However, since the development of capacity-approaching error-correcting codes, these conventional algorithms often become unreliable: state-of-the-art codes operate at very low signal-to-noise ratios (SNR), making accurate estimation very hard. A challenging problem is thus the following: how can we exploit the presence of the error-correcting code during the estimation process?

Similarly, estimation of the transmit mode needs to be performed (mode detection). This is a relatively recent problem, not present in mono-mode receivers. Mode detection is different from channel estimation in a sense that the transmitter always has perfect knowledge of the current transmit mode. This implies that when no reliable mode detection algorithm can be developed at a reasonable computational cost, it makes more sense to forward the current transmit mode to the receiver through some separate channel. When mode changes are frequent, mode detection is more attractive [27].

In Part III, we will describe a general framework for estimating all these parameters. The emphasis is placed on so-called code-aided (or code-aware) estimation techniques that iterate between data detection and estimation.

2.6 Main Points

We have presented the transmitter and receiver model, including the channel model and RF to IF to BB down-conversion. While the transmitter's task is generally straightforward, the receiver faces several challenges: it needs to detect the data stream in the presence of unknown parameters. These include parameters of the quasi-static channel, noise statistics and the transmit modes. Even when all these parameters are known to the receiver, data detection is still a hard task and should be performed in an efficient manner. As an additional challenge, the data will be protected using a powerful error-correcting code. While such codes result in more bandwidth- and/or power-efficient communication, they force the system designer to consider more advanced estimation and detection algorithms.

Chapter 3

Factor Graphs

3.1 Introduction

Factor graphs are currently a hot topic in communications research. The key concepts have been around for some time in fields such as machine learning and artificial intelligence [28]. Only recently connections have been made with communications problems [8, 9, 29]. Currently, factor graphs serve as a means to develop communications systems and algorithms [30]: many existing detection algorithms can be interpreted as special cases of factor graphs and many novel algorithms have been designed according to the factor graph paradigm. Factor graphs will appear throughout this dissertation. Since factor graphs define how algorithms can be developed in a systematic way, many of the techniques that will be investigated are based on factor graphs or rely on them for their proper operation. In any case, they are of sufficient importance to deserve their own chapter in this dissertation.

This chapter is organized as follows: we start with a description of factor graphs and the sum-product algorithm from an abstract point of view in section 3.2. We then move towards a specific application in section 3.3, common to many communications problems: maximum a posteriori (MAP) estimation. We end in section 3.4 with some examples of MAP estimation, intended to give the reader some feeling of how factor graphs can be applied in advanced detection and decoding schemes.

3.2 Factor graphs and the sum-product algorithm

Factor graphs are a convenient way to represent functions of many variables. When a function can be written as a product of (more simple) functions, the corresponding factor graph can be decomposed in multiple (more simple) factor graphs which are interconnected. Factor graphs also provide an efficient way to compute *marginals* of the corresponding function. This is achieved by a procedure, known as the *sum-product algorithm*. The marginals are computed by passing *messages* over the factor graph. We will give a brief overview of the concept of factor graphs and apply them to some well-known communications problems.

3.2.1 Factor graphs: representation and terminology

We will use the normal graphs that were introduced in [29]. A factor graph is a diagram that represents the factorization of a function of several variables:

$$f\left(x_1, x_2, \ldots, x_N\right) = \prod_j f_j\left(X_j\right) \tag{3.1}$$

where X_j is a subset of $\{x_1, x_2, \ldots, x_N\}$. Each of the variables x_i (which can be scalars or vectors) is defined over some alphabet Ψ_i, so that $f(.)$ is defined over $\Psi_1 \times \ldots \times \Psi_N$. A factor graph consists of nodes (vertices), edges and half-edges (the latter are connected to only one node). The factor graph is related to the function $f(.)$ as follows: there is a node[1] for every factor $f_j(.)$ and one (half-) edge for every variable x_k. Node f_j is connected to variable $x_k \iff x_k \in X_j$. Finally, edges (resp. half-edges) are connected to exactly two (resp. one) nodes. We also introduce the notion of the so-called *indicator function*, $I[P]$: for a predicate P, $I[P] = 0$ if P is false and $I[P] = 1$ if P is true. An *equality-node* with adjacent edges $x_1, \ldots, x_L$ is a node representing the function[2] $I[x_1 = \ldots = x_L]$.

[1]Nodes me be of different shapes (rectangles, circles, etc.). The shapes bear no meaning.

[2]In case the variables are defined over a continuous domain: $I[x_1 = \ldots = x_L] = \prod_{k=1}^{L-1} \delta(x_k - x_{k+1})$, where $\delta(x)$ is the Dirac distribution.

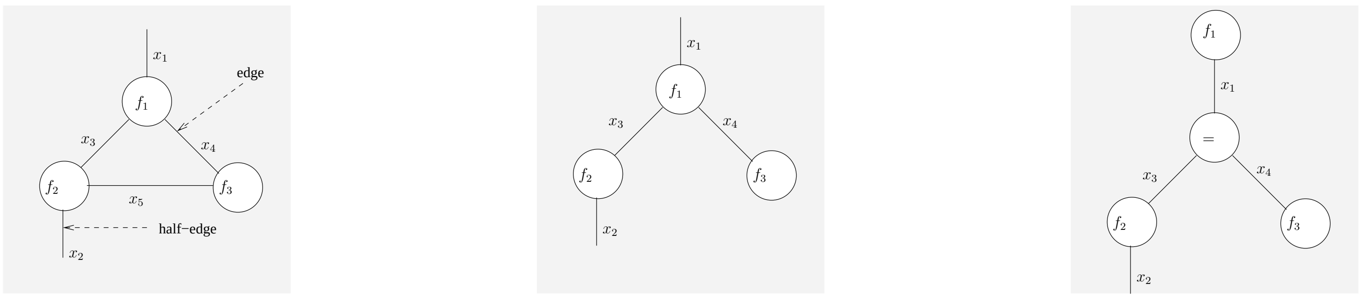


Figure 3.1: *Some simple factor graphs. Observe a cycle in the factor graph on the left*

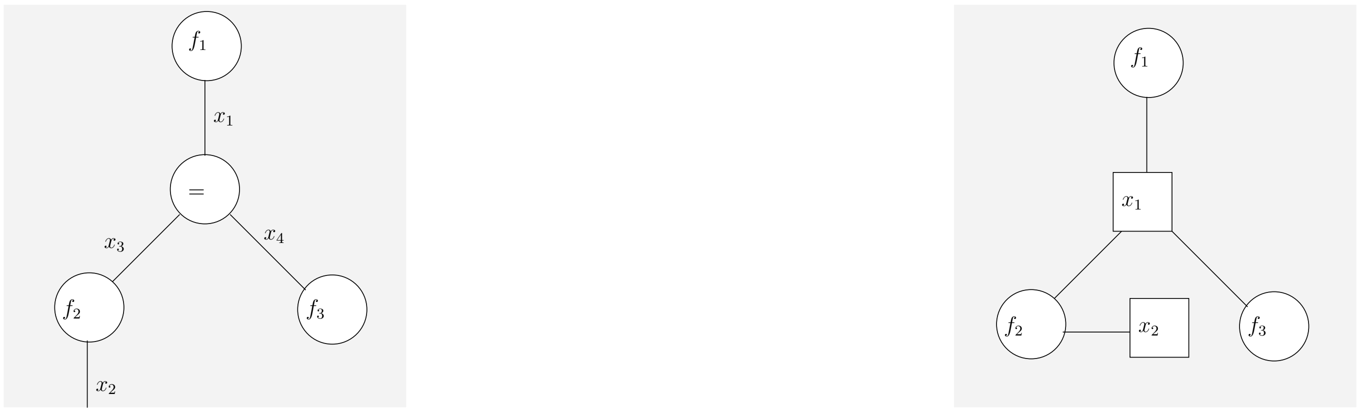


Figure 3.2: *Normal factor graphs vs. conventional-style factor graphs. The two graphs represent the same function. On the left a normal factor graph, on the right a conventional factor graph.*

Examples

In Fig. 3.1, some simple factor graphs are shown. The graph on the left corresponds to the function:

$$f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_3, x_4)\, f_2(x_2, x_3, x_5)\, f_3(x_4, x_5)\,. \tag{3.2}$$

The graph in the middle corresponds to the function:

$$f(x_1, x_2, x_3, x_4) = f_1(x_1, x_3, x_4)\, f_2(x_2, x_3)\, f_3(x_4)\,. \tag{3.3}$$

The reader will observe that the graph on the left contains a cycle (of length 3), while the other graphs do not. Also note that variables can occur in no more than two functions. Although this may seem restrictive, it is not really: by including equality nodes, variables can appear in arbitrarily many functions. For instance, the graph on the right represents the function:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= f_1(x_1)\, f_2(x_2, x_3)\, f_3(x_4)\, I[x_1 = x_3 = x_4] && (3.4)\\ &= f_1(x_1)\, f_2(x_2, x_1)\, f_3(x_1)\, I[x_1 = x_3 = x_4] && (3.5) \end{aligned}$$

so that x_3 and x_4 can be interpreted as 'dummy' variables in the factor graph representation of the function $f_1(x_1)\, f_2(x_2, x_1)\, f_3($

Representation

In technical literature, factor graphs are conventionally depicted in a slightly different way. In [8], both variables and functions correspond to nodes, while the edges define which variable appears in which function. For instance, the function (3.5) leads to a conventional factor graph of the form shown on the right of Fig. 3.2. The corresponding normal factor graph is depicted on the left side of the same figure. We will only consider normal factor graphs in this dissertation.

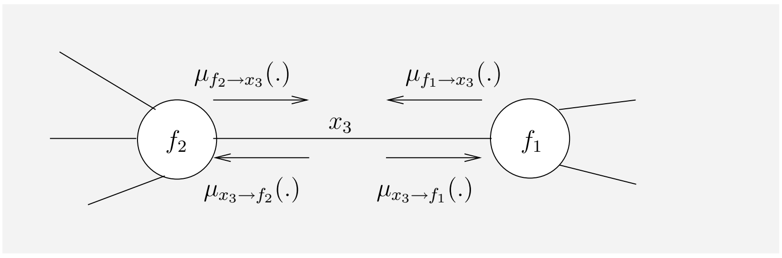


Figure 3.3: *Messages on factor graphs. The message $\mu_{f_2\to x_3}(.)$ is passed over edge x_3 where it is renamed $\mu_{x_3\to f_1}(.)$. Similarly, $\mu_{f_1\to x_3}(.) = \mu_{x_3\to f_2}(.)$.*

3.2.2 The sum-product algorithm

Factor graphs not only allow us to visualize a function, they can also provide a graphical way to compute its *marginals*, $g_i(x_i)$, for $x_i \in \Psi_i$, defined as

$$g_i(x_i) = \sum_{\sim\{x_i\}} f(x_1, x_2, \ldots, x_N) \tag{3.6}$$

where $\sim\{x_i\}$ represents the 'not-sum', i.e., the summation over *all* variables, *except* x_i. In (3.6), summations over continuous valued variables must be replaced with integrals over the corresponding domain.

In many cases, the computation of $g_i(x_i)$ and of $g_j(x_j)$, $j \neq i$, will have a lot of commonality. To compute all the marginals jointly in an efficient manner, we resort to the sum-product (SP) algorithm. The SP algorithm is a *message-passing* algorithm, where messages are computed in the nodes of the graph and passed over the edges. A message over a given edge is a *function* of the variable corresponding to that edge. The message from a given node (i.e., a function f_m) to one of its adjacent edges (i.e., a variable of that function, $x_i \in X_m$) is denoted by $\mu_{f_m\to x_i}(.)$. Similarly, a message from an edge (variable x_j) to an adjacent function f_n is denoted by $\mu_{x_j\to f_n}(.)$. How different messages are related is depicted in Fig. 3.3 (note that messages may be renamed as they pass over an edge).

The marginal $g_i(.)$, evaluated in the value $x_i \in \Psi_i$, is given by the product of two messages over the corresponding edge.

$$g_i(x_i) = \mu_{f_m\to x_i}(x_i) \times \mu_{x_i\to f_m}(x_i) \tag{3.7}$$

where f_m is an arbitrary function with $x_i \in X_m$. The key to the SP algorithm is how messages in nodes are computed.

Theorem 3.2.1 (The Sum-Product algorithm). *Given a function and a cycle-less corresponding factor graph representation, the Sum-Product Algorithm relates outgoing messages to incoming messages, according to*[3]

$$\mu_{f_n\to x_k}(x_k) = \sum_{\sim\{x_k\}} f_n(X_n) \prod_{l\neq k} \mu_{x_l\to f_n}(x_l). \tag{3.8}$$

Initialization: *The SP algorithm starts from the half-edges in the graph and from nodes with degree*[4] *1. Half-edges transmit messages identically equal to '1', while degree-1 nodes transmit messages equal to the corresponding function itself, evaluated in the connected edge.*
Message Computation: *A node $f_n(.)$ of degree m with corresponding variables $x_1, \ldots, x_m$ computes an outgoing message $\mu_{f_n\to x_k}(x_k)$ on edge x_k according to (3.8) when all incoming messages ($\mu_{x_l\to f_n}(x_l)$, $l \neq k$) have been received.*
Termination: *Once all messages have been computed, the marginals are given by (3.7).*

A sketch of the proof is given in the Appendix of this chapter.

3.2.2.1 Remarks

The sum-product algorithm is more general than one would suppose at first sight: the sum and product operators need not be the standard sum and product over the set of real numbers. The sum-product algorithm can be applied to general abstract sets F, endowed with suitable 'sum' ($\oplus$) and 'product' ($\otimes$) operations, such that $(F, \oplus, \otimes)$ forms a commutative semi-ring [9]:

- $\oplus$ is associative and commutative. There exists an identity element for $\oplus$: $e_\oplus$

[3]For continuous variables, the summations are replaced with integrals.
[4]The degree of a node is the number of adjacent edges.

- $\otimes$ is associative and commutative. There exists an identity element for $\otimes$, $e_\otimes$
- $\otimes$ is distributive over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ for any $a, b, c \in F$.

The indicator function is then given by : $I[P] = e_\otimes$ when P is true and $I[P] = e_\oplus$ when P is false. Also, in the initialization step, half-edges transmit the message $e_\otimes$.

Observe also that the sum-product algorithm is used to find marginals of the function. These marginals are functions themselves. Similarly, the messages are functions of the corresponding edges.

3.2.2.2 Example

Consider the function from Eq. (3.5): $f(x_1, x_2) = f_1(x_1) f_2(x_2, x_1) f_3(x_1)$ with corresponding factor graph shown in Fig. 3.4. The sum-product algorithm is applied as follows:

1. **Initialization:** Messages from half-edges and degree-1 nodes:

$$\begin{cases} \mu_{x_2 \to f_2}(x_2) &= 1 \\ \mu_{f_3 \to x_4}(x_4) &= f_3(x_4) \\ \mu_{f_1 \to x_1}(x_1) &= f_1(x_1) \end{cases}$$

2. Now, messages can be computed from f_2 to edge x_3 and from the equality node to x_3:

$$\begin{aligned} \mu_{f_2 \to x_3}(x_3) &= \sum_{x_2} f_2(x_2, x_3)\, \mu_{x_2 \to f_2}(x_2) \\ &= \sum_{x_2} f_2(x_2, x_3) \\ \mu_{\boxed{=} \to x_3}(x_3) &= \sum_{x_1, x_4} I[x_1 = x_3 = x_4]\, \mu_{f_3 \to x_4}(x_4)\, \mu_{f_1 \to x_1}(x_1) \\ &= \mu_{f_3 \to x_4}(x_3)\, \mu_{f_1 \to x_1}(x_3) \end{aligned}$$

3. Subsequently, we compute messages from the equality node to x_1 and x_4; at the same time a message from f_2 to x_2 can be computed:

$$\begin{aligned} \mu_{\boxed{=} \to x_1}(x_1) &= \mu_{f_2 \to x_3}(x_1)\, \mu_{f_3 \to x_4}(x_1) \\ \mu_{\boxed{=} \to x_4}(x_4) &= \mu_{f_2 \to x_3}(x_4)\, \mu_{f_1 \to x_1}(x_4) \\ \mu_{f_2 \to x_2}(x_2) &= \sum_{x_3} f_2(x_2, x_3)\, \mu_{\boxed{=} \to x_3}(x_3) \end{aligned}$$

4. **Termination:** Finally, the marginals are given by:

$$\begin{aligned} g_1(x_1) &= \mu_{\boxed{=} \to x_1}(x_1) \times \mu_{f_1 \to x_1}(x_1) \\ &= \sum_{x_2} f_2(x_2, x_1)\, f_3(x_1)\, f_1(x_1) \\ g_2(x_2) &= \mu_{f_2 \to x_2}(x_2) \times \mu_{x_2 \to f_2}(x_2) \\ &= \sum_{x_3} f_2(x_2, x_3)\, f_3(x_3)\, f_1(x_3) \end{aligned}$$

   which is clearly the desired result. Also, it is easily verified that $g_1(x_1 = a) = g_3(x_3 = a) = g_4(x_4 = a)$ for any $a \in \Psi_1$.

3.2.3 Factor graphs with cycles

When the graph contains cycles (loops), the SP algorithm has no natural initialization, nor termination. The SP algorithm starts again from the half-edges and nodes with degree one. At some point, due to cyclic dependencies, nodes will not be able to compute outgoing messages. The sum-product algorithm simply halts (or waits forever). The common solution to this problem is to replace unknown messages with a 'default' message (the message '1' ($e_\otimes$ in general) over the corresponding domain). This is known as *resetting* the factor graph. The SP algorithm keeps operating in this way. After a number of iterations, the SP algorithm is halted. The computed marginals are no longer exact, but rather an approximation. The approximation becomes less accurate when the graph contains short cycles [8].

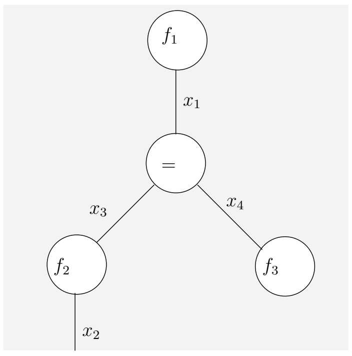


Figure 3.4: *A factor graph of the function* $f(x_1, x_2) = f_1(x_1) f_2(x_2, x_1) f_3(x_1)$ *with two dummy variables* x_3 *and* x_4.

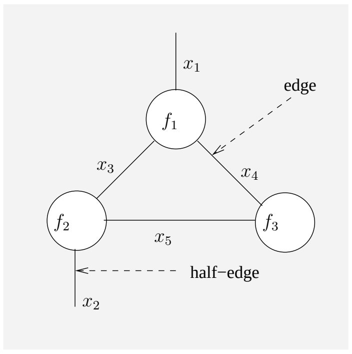


Figure 3.5: *A factor graph of the function* $f_1(x_1, x_3, x_4) f_2(x_2, x_3, x_5) f_3(x_4, x_5)$ *with a cycle of length 3.*

3.2.3.1 Example

Consider the function from Eq. (3.2): $f_1(x_1, x_3, x_4) f_2(x_2, x_3, x_5) f_3(x_4, x_5)$ with corresponding factor graph shown in Fig. 3.5. The sum-product algorithm is applied as follows:

1. **Initialization:** Messages from half-edges and degree-1 nodes:

$$\begin{cases} \mu_{x_1 \to f_1}(x_1) & = & 1 \\ \mu_{x_2 \to f_2}(x_2) & = & 1 \end{cases}$$

2. At this point, no more messages can be computed. For instance, to compute $\mu_{f_2 \to x_3}(x_3)$, we require $\mu_{x_5 \to f_2}(x_5)$. Since $\mu_{x_5 \to f_2}(x_5)$ is unavailable, we set $\mu_{x_5 \to f_2}(x_5) = 1$.

3. Now, the message from f_2 to x_3 can be computed:

$$\mu_{f_2 \to x_3}(x_3) = \sum_{x_2, x_5} f_2(x_2, x_3, x_5) \times 1 \times 1$$

4. We can then compute $\mu_{f_1 \to x_4}(x_4)$ and $\mu_{f_3 \to x_5}(x_5)$, to re-evaluate $\mu_{f_2 \to x_3}(x_3)$, to re-compute $\mu_{f_1 \to x_4}(x_4)$ and $\mu_{f_3 \to x_5}(x_5)$,... ad infinitum.

5. **Termination**: since this process continues indefinitely, we simply interrupt the SPA and compute all approximate marginals.

3.3 Receivers as factor graphs

It turns out that many communication problems can be cast in the framework of factor graphs. In fact, some state-of-the-art iterative algorithms that were originally developed in an ad-hoc way, can be interpreted as applying the sum-product algorithm on a suitable factor graph.

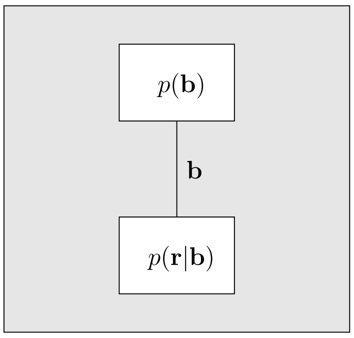


Figure 3.6: *Factor graph of* $p(\mathbf{r}|\mathbf{b})\,p(\mathbf{b})$*. The only variable (edge) is* $\mathbf{b}$.

3.3.1 Principle

Suppose we want to infer $\mathbf{b}$ from an observation $\mathbf{r}$. With $\mathbf{b} = [b_0, \dots, b_{N_b-1}]^T$, we commonly want to minimize the error probability $P\left[\hat{\mathbf{b}} \neq \mathbf{b}\right]$ by maximizing the *a posteriori distribution* w.r.t. $\mathbf{b}$ [16], resulting in the MAP[5] estimate of $\mathbf{b}$:

$$\hat{\mathbf{b}}_{MAP} \quad = \quad \arg\max_{\mathbf{b}} p(\mathbf{b}|\mathbf{r}). \tag{3.9}$$

In communications problems, the rule (3.9) corresponds to MAP sequence detection. In most cases, direct evaluation of $p(\mathbf{b}|\mathbf{r})$ is impossible. Let us re-write the a posteriori distribution using Bayes' Rule[6]:

$$p(\mathbf{b}|\mathbf{r}) \quad \propto \quad p(\mathbf{r}|\mathbf{b})\,p(\mathbf{b}) \tag{3.10}$$

where $a \propto b$ bears the meaning: a is equal to b, up to irrelevant additive and multiplicative constants. This factorization can be represented by the factor graph from Fig. 3.6. The observation $\mathbf{r}$ should be considered a parameter (rather than a variable) in this factor graph. Applying the sum-product algorithm to this graph would result in the computation of $p(\mathbf{b}|\mathbf{r})$ (up to an irrelevant multiplicative constant). Unfortunately, even the *likelihood function* $p(\mathbf{r}|\mathbf{b})$ and especially the *a priori distribution* $p(\mathbf{b})$ are commonly too complex to evaluate.

Let us transform this factor graph into a graph where the variables are now the *components* of $\mathbf{b}$. The corresponding factor graph is depicted in Fig. 3.7. Although the latter graph represents the same function as the graph from Fig. 3.6, the SP algorithm will not yield the same results: in the case of Fig. 3.7, we can compute the *marginal* a posteriori distributions $p(b_k|\mathbf{r})$, not the *joint* a posteriori distribution $p(\mathbf{b}|\mathbf{r})$. Decisions with respect to the components of $\mathbf{b}$ are given by:

$$\hat{b}_{k,MAP} = \arg\max_{b_k} p(b_k|\mathbf{r}). \tag{3.11}$$

In communications problems, the rule (3.11) corresponds to MAP symbol detection. Generally, grouping the MAP estimates of the components does not yield the MAP estimate of the vector $\mathbf{b}$:

$$\hat{\mathbf{b}}_{MAP} \neq \left[\hat{b}_{0,MAP}, \dots, \hat{b}_{N_b-1,MAP}\right]^T. \tag{3.12}$$

Nevertheless, in many practical scenarios, computation of the marginal $p(b_k|\mathbf{r})$ is feasible, provided we introduce additional variables (edges). Furthermore, the technique of making decisions w.r.t. the components of $\mathbf{b}$ as opposed to $\mathbf{b}$ as a whole has led to some of the most powerful detection and decoding algorithms known today. The idea is to introduce additional variables to the likelihood function $p(\mathbf{r}|\mathbf{b})$ and the a priori distribution $p(\mathbf{b})$ so that both lead to nice factorizations. This leads to the following

Key Idea: *In order to perform MAP detection on the components of* $\mathbf{b}$*, given an observation* $\mathbf{r}$*, we need to compute the a posteriori probabilities (APPs)* $p(b_k|\mathbf{r})$*, i.e., the marginals of* $p(\mathbf{b}|\mathbf{r})$*. We introduce additional variables (grouped into a vector, say* $\mathbf{x}$*), such that* $p(\mathbf{b},\mathbf{x}|\mathbf{r})$ *leads to a convenient factor graph representation of both the likelihood function* $p(\mathbf{r}|\mathbf{b},\mathbf{x})$ *and the a priori distribution* $p(\mathbf{b},\mathbf{x})$*. Applying the SP algorithm on this graph yields the required APPs* $\{p(b_k|\mathbf{r})\}$*, as well as APPs of the additional variables* $\{p(x_l|\mathbf{r})\}$.

[5]MAP: Maximum A Posteriori.
[6]Bayes' Rule: $p(x,y) = p(x|y)\,p(y) = p(y|x)\,p(x)$.

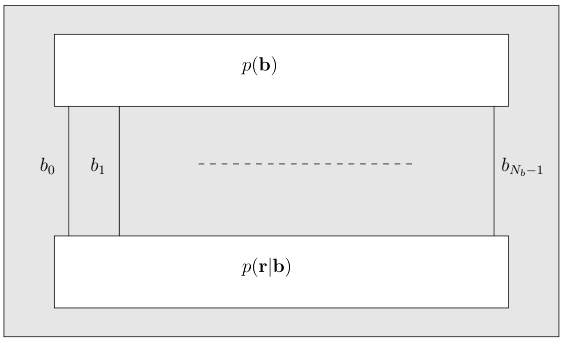


Figure 3.7: *Another factor graph of* $p(\mathbf{r}|\mathbf{b})\,p(\mathbf{b})$. *The variables are now the components of* $\mathbf{b}$.

To be more precise, these graphs will represent a function that is proportional to an a posteriori distribution. Hence, the marginals will be marginal a posteriori distributions, up to a scaling factor. Since distributions are normalized, we can easily determine this scaling factor. As will become apparent, this implies that messages in the SP algorithm can be scaled arbitrarily: this does not affect the outcome of the SP algorithm (it merely changes the final scaling factor). In the following sections, we will often scale messages implicitely, or determine messages up to a scaling factor (through the notation $a \propto b$).

3.3.2 Sequence detection

In some situations, it may be required to find the MAP *sequence estimate* (i.e., $\hat{\mathbf{b}}_{MAP}$), rather than the MAP estimates of the individual components ($\left\{\hat{b}_{k,MAP}\right\}$). For instance, for convolutional codes, the well-known Viterbi algorithm [31] locates the MAP sequence estimate. It is reasonable to ask ourselves, can an algorithm such as the Viterbi algorithm be cast within the factor graphs framework? Indeed it can. The idea is as follows: first of all, note that $(F, \oplus, \otimes) = (\mathbb{R}^+, \max, \times)$ forms a commutative semi-ring. Now, we introduce additional variables, such that $p(\mathbf{b}, \mathbf{x}|\mathbf{r})$ has a convenient factorization. We then construct the corresponding factor graph. When we apply the SP algorithm (where now the 'sum'-operation is replaced by max), we find the following marginals (up to an irrelevant constant):

$$g_k(b_k) = \max_{\mathbf{b},\mathbf{x}:b_k} p(\mathbf{b}, \mathbf{x}|\mathbf{r}). \tag{3.13}$$

Introducing

$$\hat{b}_k = \arg\max_{b_k} g_k(b_k) \tag{3.14}$$

we finally obtain

$$\hat{\mathbf{b}}_{MAP} = \left[\hat{b}_0, \ldots, \hat{b}_{N_b-1}\right]^T. \tag{3.15}$$

Commonly, this algorithm is executed in the log-domain, where $(F, \oplus, \otimes) = (\mathbb{R}, \max, +)$. In that case, we marginalize the factorization[7] of $\log p(\mathbf{b}, \mathbf{x}|\mathbf{r})$.

3.4 Examples: codes on graphs

In this section, we will investigate some important examples:

- a generic error-correcting code,
- mapping of bits to constellation points,
- a convolutional code,
- a turbo code.

We will create factor graph descriptions of each of these systems.

[7]Note that the factorization now is no longer a product of factors, but a sum of terms!

3.4.1 Some basic building blocks

Interleaver

An interleaver is a block that permutes edges (variables). The inverse process is known as de-interleaving. Although an interleaver can be written as a function with a corresponding factor graph, we choose to interpret it merely as a re-ordering of edges. Hence, an interleaver has no real factor graph representation, it is simply a means to interconnect factor graphs. An example is shown in Fig. 3.8.

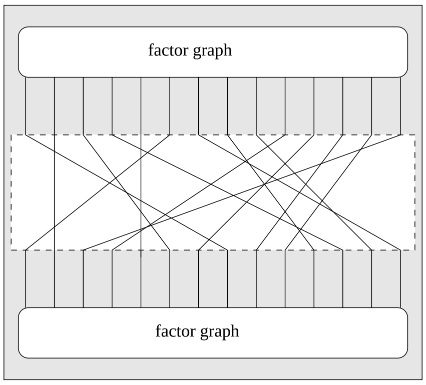


Figure 3.8: *An interleaver connecting two factor graphs*

Copier

A copier is an extension of the equality node, so that the attached variables can appear in multiple functions. An example of a 3×3 copier is shown in Fig. 3.9. The sum-product algorithm is trivial: an outgoing message is the product of the two incoming messages attached to the corresponding equality node. For instance, a copier which copies a variable a to a_1 and a_2 leads to

$$\begin{aligned} \mu_{\boxed{=}\to a}(a) &= \sum_{a_1,a_2} I\left[a_1 = a_2 = a\right] \mu_{a_1 \to \boxed{=}}(a_1)\, \mu_{a_2 \to \boxed{=}}(a_2) \\ &= \mu_{a_1 \to \boxed{=}}(a)\, \mu_{a_2 \to \boxed{=}}(a)\,. \end{aligned}$$

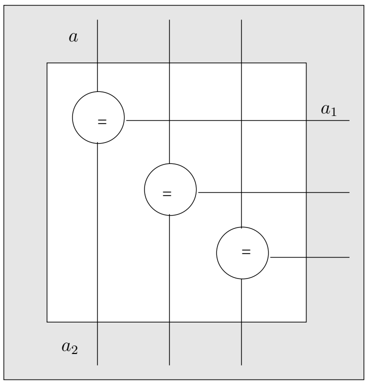


Figure 3.9: *A 3×3 Copier*

Selector

A selector (or puncturer) has L_1 inputs and L_2 outputs, with $L_2 \leq L_1$. The output is a subset of the input. A selector can again be considered as a collection of edges, and serves to connect factor graphs. A graph is shown in Fig. 3.10. The arrow in the graph represents the direction of the operation of the selector. The L_1 edges at the top and the L_2

edges at the bottom are connected to other nodes in the factor graph. The $L_1 - L_2$ edges that end in the selector-block are half-edges.

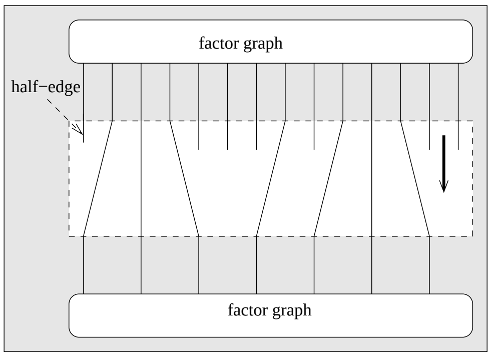


Figure 3.10: *A Selector connecting two factor graphs.*

3.4.2 A generic error-correcting code

3.4.2.1 Definition

A binary error-correcting code of length N_c is an injection from $\{0,1\}^{N_b} \to \{0,1\}^{N_c}$, with $N_c \geq N_b$. The ratio N_b/N_c is referred to as the code rate. Consider a binary code of length N_c with 2^{N_b} codewords. A codeword $\mathbf{c} \in \{0,1\}^{N_c}$ is obtained by mapping an information word $\mathbf{b} \in \{0,1\}^{N_b}$, such that $\mathbf{c} = \chi(\mathbf{b})$. Let us further assume that the codeword $\mathbf{c}$ is mapped to a sequence of N_c BPSK symbols[8] $\mathbf{a} = 2\mathbf{c} - \mathbf{1}$. Finally, $\mathbf{a}$ is transmitted over a discrete-time AWGN channel. At the receiver, we have an observation $\mathbf{r} = \mathbf{a} + \mathbf{n}$, $\mathbf{n}$ is a vector of N_c iid (independent and identically distributed) AWGN samples with $n_k \sim \mathcal{N}(0, \sigma^2)$. The goal of the receiver is to recover the information stream $\mathbf{b}$. The MAP symbol detection rule (3.11) leads to:

$$\hat{b}_{k,MAP} = \arg\max_{b_k} p(b_k | \mathbf{r}) \tag{3.16}$$

$$= \arg\max_{b_k} \sum_{\sim\{b_k\}} p(\mathbf{b} | \mathbf{r}) \tag{3.17}$$

where

$$p(\mathbf{b} | \mathbf{r}) = \sum_{\mathbf{c},\mathbf{a}} p(\mathbf{b}, \mathbf{a}, \mathbf{c} | \mathbf{r}). \tag{3.18}$$

Now, $p(\mathbf{b}, \mathbf{a}, \mathbf{c} | \mathbf{r})$ can be factorized as follows:

$$p(\mathbf{b}, \mathbf{a}, \mathbf{c} | \mathbf{r}) \propto p(\mathbf{b}) \prod_{k=0}^{N_c-1} p(r_k | a_k) I[a_k = 2c_k - 1] I[\mathbf{c} = \chi(\mathbf{b})] \tag{3.19}$$

where, due to the AWNG, $p(r_k | a_k) \propto \exp\left(-\frac{1}{2\sigma^2} |r_k - a_k|^2\right)$.

3.4.2.2 Factor graph

The factor graph of the factorization (3.19) is shown in Fig. 3.11. The top-most node corresponds to $p(\mathbf{b})$, the a priori distribution of $\mathbf{b}$. This node may be omitted when all bits b_k are mutually independent with uniform a priori distributions[9]. The node below enforces the code constraints $I[\mathbf{c} = \chi(\mathbf{b})]$ between $\mathbf{b}$ and $\mathbf{c}$. The nodes marked φ enforce the mapping constraint (mapper nodes) and the bottommost nodes correspond to $p_k = p(r_k | a_k)$. Observe that there are cycles between the node $p(\mathbf{b})$ and the node $I[\mathbf{c} = \chi(\mathbf{b})]$.

[8] $\Omega = \{-1, +1\}$. More general mapping strategies will be covered later.
[9] This is because the resulting half-edges will transmit uniform messages.

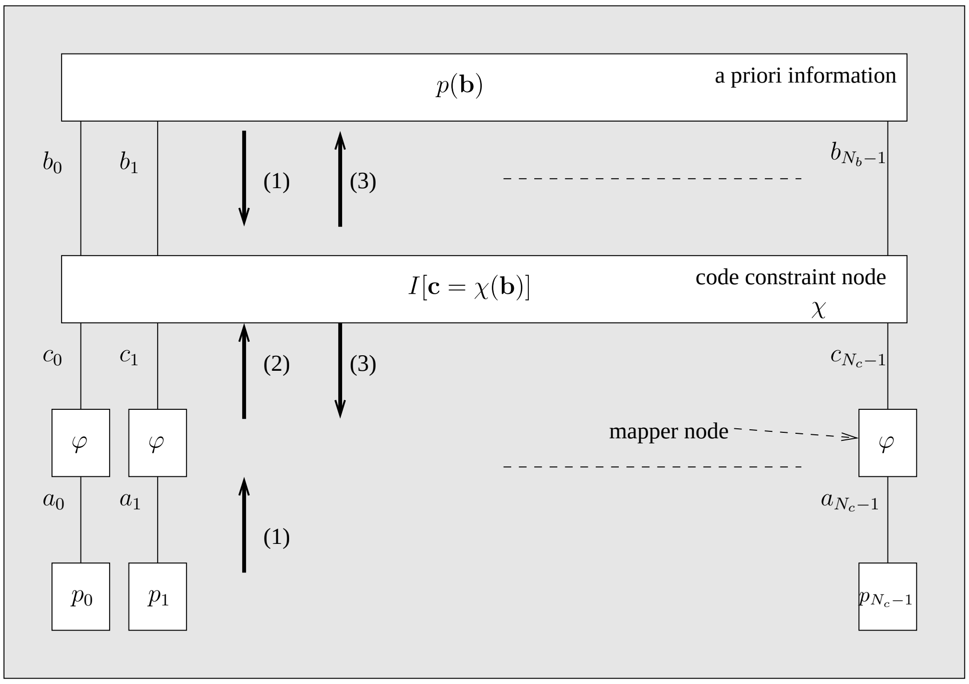


Figure 3.11: *Factor graph of the function* $p(\mathbf{b}) \prod_{k=0}^{N_c-1} p(r_k | a_k) I[a_k = 2c_k - 1] I[\mathbf{c} = \chi(\mathbf{b})]$ *of a generic code with BPSK mapping. The nodes* φ *and* χ *enforce the constraint* $I[a_k = 2c_k - 1]$ *and* $I[\mathbf{c} = \chi(\mathbf{b})]$, *respectively. The bold arrows represent the scheduling of the messages.*

3.4.2.3 Sum-product algorithm

Application of the sum-product algorithm will yield the (approximate) marginal APPs $p(b_k | \mathbf{r})$, $k = 0, \ldots, N_b - 1$, as well as the by-products $p(c_k | \mathbf{r})$ and $p(a_k | \mathbf{r})$ for $k = 0, \ldots, N_c - 1$. To make the situation a little more insightful, let us assume that the information bits b_k are iid[10] with $p(b_k) = 1/2, \forall k$. In order to decode this code, we now apply the sum-product algorithm (the ordering of messages in steps 1-3 is depicted in Fig. 3.11.):

1. **Initialization:** Messages from the N_c bottom-most nodes to the mapper-nodes are sent:

$$\mu_{a_n \to \varphi}(a_n) \propto p(r_n | a_n)$$

   and at the same time messages from the top-most edges to the block enforcing the code constraints

$$\mu_{b_k \to \chi}(b_k) = 1/2$$

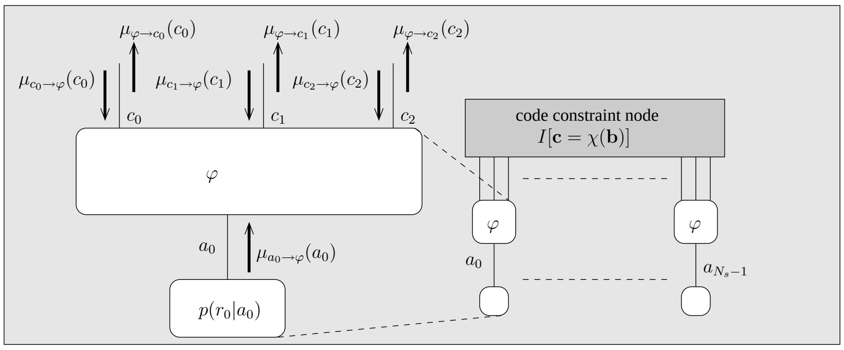
2. Messages from the N_c mapper-nodes φ to the code constraint node $I[\mathbf{c} = \chi(\mathbf{b})]$ are given by, applying (3.8)

$$\begin{aligned} \mu_{\varphi \to c_n}(c_n) &= \sum_{a_n} I[a_n = 2c_n - 1] \mu_{a_n \to \varphi}(2c_n - 1) \\ &= \mu_{a_n \to \varphi}(2c_n - 1) \end{aligned}$$

3. Now the messages from code constraint block $I[\mathbf{c} = \chi(\mathbf{b})]$ to the mapper nodes φ can be computed according to (3.8)

$$\mu_{\chi \to c_m}(c_m) = \tag{3.20}$$
$$\sum_{\mathbf{b},\mathbf{c}:c_m} I[\mathbf{c} = \chi(\mathbf{b})] \prod_{k=0}^{N_b-1} \mu_{b_k \to \chi}(b_k) \prod_{n=0,n\neq m}^{N_c-1} \mu_{\varphi \to c_n}(c_n)$$

[10]Note that this assumption removes the cycles present in the factor graph from Fig. 3.11: the node $p(\mathbf{b})$ can be replaced with N_b nodes $p(b_k)$. When $p(b_k)$ is uniform, the nodes $p(b_k)$ can themselves be removed, leaving us with N_b half-edges b_k, $k = 0, \ldots, N_b - 1$.



$\mu_{\varphi\to c_0}(c_0)$
$\mu_{\varphi\to c_1}(c_1)$
$\mu_{\varphi\to c_2}(c_2)$
$\mu_{c_0\to\varphi}(c_0)$
$\mu_{c_1\to\varphi}(c_1)$
$\mu_{c_2\to\varphi}(c_2)$
c_0
c_1
c_2
code constraint node
$I[\mathbf{c}=\chi(\mathbf{b})]$
φ
a_0
$\mu_{a_0\to\varphi}(a_0)$
$p(r_0|a_0)$
a_{N_s-1}

Figure 3.12: *Mapper factor graph for $m = 3$ bits*

where the summation goes over all 2^{N_b} possible $\mathbf{b}$ and all 2^{N_c-1} possible $\mathbf{c}$ with m-th bit equal to c_m. At the same time, messages are sent from the code constraint block to the information bit edges

$$\mu_{\chi\to b_l}(b_l) = \sum_{\mathbf{c},\mathbf{b}:b_l} I[\mathbf{c} = \chi(\mathbf{b})] \prod_{k=0,k\neq l}^{N_b-1} \mu_{b_k\to\chi}(b_k) \prod_{n=0}^{N_c-1} \mu_{\varphi\to c_n}(c_n) \tag{3.21}$$

where the summation goes over all 2^{N_c} possible $\mathbf{c}$ and all 2^{N_b-1} possible $\mathbf{b}$ with l-th bit equal to b_l.

4. **Termination:** The APPs of the information bits are then given exactly by

$$p(b_k|\mathbf{r}) \propto \mu_{\chi\to b_k}(b_k) \times \mu_{b_k\to\chi}(b_k)$$

and the decisions of the bits can be made as follows:

$$\hat{b}_k = \arg\max_{b_k\in\{0,1\}} p(b_k|\mathbf{r}).$$

Remarks

- The power of many advanced codes lies in the efficient computation of (3.20) and (3.21), based on a factorization of $I[\mathbf{c} = \chi(\mathbf{b})]$, through the introduction of additional variables.
- As mentioned before, when the components of $\mathbf{b}$ are a priori independent, the block corresponding to $p(\mathbf{b})$ is to be replaced with N_b blocks $p(b_k)$. When $p(b_k)$ is uniform, the corresponding a priori node may be omitted all-together.
- When the components of $\mathbf{b}$ are not a priori independent, one would then have to iterate between the a priori node $p(\mathbf{b})$ and the code-constraint node $I[\mathbf{c} = \chi(\mathbf{b})]$.
- The factor graph from Fig. 3.11 can be interpreted as consisting of two parts: one part corresponding to the constraints (a priori distribution, code constraints, mapping constraints) and a second part corresponding to the observation (the N_s nodes at the bottom).

3.4.3 Mapper

In the previous case, coded bits were mapped to BPSK symbols. We will now consider a more general scenario. The sequence of coded bits $\mathbf{c} = [c_0, \ldots, c_{N_c-1}]^T$ is broken up into $N_s = N_c/m$ blocks of length m. Let us denote the l-th block as $\mathbf{c}_l$. A mapper converts each group of m bits to a complex number, part of a 2^m-point signaling constellation Ω. So, if we want to transmit m bits $\mathbf{c}_l = [c_{lm}, \ldots, c_{(l+1)m-1}]^T$ over an AWGN channel using a constellation Ω, we map these bits to a point $a_l = \varphi(\mathbf{c}_l) \in \Omega$. Transmission of a_l over the AWGN channel yields $r_l = a_l + n_l$, with n_l a complex AWGN sample with variance σ^2 per real dimension. Hence, a mapper corresponds to the function $I[a_l = \varphi(\mathbf{c}_l)]$, and the overall mapping function[11] $I[\mathbf{a} = \varphi(\mathbf{c})]$ can be factorized

$$I[\mathbf{a} = \varphi(\mathbf{c})] = \prod_{l=0}^{N_s-1} I[a_l = \varphi(\mathbf{c}_l)]. \tag{3.22}$$

[11]With a slight abuse of the notation φ.

A factor graph for this mapping function is depicted in Fig. 3.12 for $m = 3$. An array of such mapper nodes is to be combined with the generic error-correcting code from the previous section. Note that there are cycles present between the mapper nodes and the code-constraint node when $m > 1$. Hence, the SP algorithm applied on such a factor graph will be iterative and sub-optimal [32]. Let us focus on the first symbol $l = 0$:

1. **Initialization:** a message from the observation node (at the bottom) to the mapper node is sent:

$$\mu_{a_0 \to \varphi}(a_0) \propto \exp\left(-\frac{1}{2\sigma^2}|r_l - a_0|^2\right).$$

   At the same time, messages from the edges of the coded bits are sent to the mapper block $\mu_{c_j \to \varphi}(c_j)$. In case such messages are unavailable, we set $\mu_{c_j \to \varphi}(c_j) = 1$, $j = 0, \ldots, m-1$.

2. Messages from the mapper node to the coded bits can now be evaluated:

$$\mu_{\varphi \to c_i}(c_i) = \sum_{\mathbf{c}:c_i} I[a_0 = \varphi(\mathbf{c}_0)]\, \mu_{a_0 \to \varphi}(a_0) \prod_{j \neq i} \mu_{c_j \to \varphi}(c_j) \tag{3.23}$$

   where the summation goes over all m-bit sequences with i-th bit equal to c_i.

3. The messages $\mu_{\varphi \to c_i}(c_i)$ can then be forwarded to the code-constraint block $I[\mathbf{c} = \chi(\mathbf{b})]$. After some time, messages $\mu_{c_j \to \varphi}(c_j)$ are provided from the code-constraint block to the mapper block, so that (3.23) can be re-evaluated.

3.4.4 Convolutional Code Generator

Similar to the generic error-correcting code, a convolutional code (CC) maps a sequence $\mathbf{b}$ of N_b information bits to a sequence $\mathbf{c}$ of N_c coded bits. Convolutional codes can be defined through a filter-like operation. For our purposes, it is more convenient to start from a state-space view. We will work in stages: first, we define a CC generator, giving rise to an indicator function $I[[\mathbf{s}, \mathbf{d}] = \chi_1(\mathbf{b})]$ with a corresponding factor graph. In the next section, further nodes will be be attached to result in a full-blown, very general convolutional code. The corresponding factor graph can then be attached to mapper nodes and observation nodes. To lighten the following exposition, we will assume again that the information bits are iid and equiprobable.

3.4.4.1 Definition

Consider a state space $\mathcal{S}$ with $2^{\nu-1}$ states. At time k, the encoder is in a given state $s_k \in \mathcal{S}$. At time k, the generator receives a number (say K) input bits $\mathbf{b}_k \in 2^K$. Given the current state s_k and the input sequence $\mathbf{b}_k$, a parity sequence $\mathbf{p}_k \in 2^M$ of length M is generated and a new state $s_{k+1} \in \mathcal{S}$ is entered into. We can write this as:

$$[\mathbf{p}_k,\, s_{k+1}] = f_{all}([\mathbf{b}_k,\, s_k]) \tag{3.24}$$

where f_{all} is independent of the time index k. Encoding starts from an initial state $s_0 = s_{start}$, where $s_{start} \in S$ is known to both the transmitter and the receiver. After L (with $N_b = KL$) input blocks $\mathbf{b}_0, \ldots, \mathbf{b}_{L-1}$, the encoding stops and the encoder is in a state s_L.

The input of a CC generator is given by $\mathbf{b} = \left[\mathbf{b}_0^T, \ldots, \mathbf{b}_{L-1}^T\right]^T$. This uniquely determines the parity sequence $\mathbf{p} = \left[\mathbf{p}_0^T, \ldots, \mathbf{p}_{L-1}^T\right]^T$. We further distinguish between non-terminated and terminated convolutional codes.

Terminated convolutional code We are required to end up in a known[12] state $s_{end} \in S$. For this purpose, we add ν termination blocks $\mathbf{t}_0, \ldots, \mathbf{t}_{\nu-1}$ so that, for $k = 0, \ldots, \nu - 1$:

$$[\mathbf{q}_k,\, s_{L+k+1}] = f_{all}([\mathbf{t}_k,\, s_{L+k}]). \tag{3.25}$$

By proper selection of $\mathbf{t}_0, \ldots, \mathbf{t}_{\nu-1}$, we can always ensure that the final state $s_{L+\nu}$ equals s_{end}. Introducing $\mathbf{t} = \left[\mathbf{t}_0^T, \ldots, \mathbf{t}_{\nu-1}^T\right]^T$ and $\mathbf{q} = \left[\mathbf{q}_0^T, \ldots, \mathbf{q}_{\nu-1}^T\right]^T$, we can write $\left[\mathbf{t}^T \mathbf{q}^T\right]^T = g_{term}(s_L)$ for some well-defined function $g_{term} : S \to \{0,1\}^{\nu \times (M+L)}$. Hence, we can write the output of the CC generator as $\mathbf{d} = \left[\mathbf{b}^T \mathbf{t}^T \mathbf{p}^T \mathbf{q}^T\right]^T$, while the corresponding state sequence is given by $\mathbf{s} = [s_0 = s_{start}, \ldots, s_{L+\nu} = s_{end}]^T$. The initial state s_{start} and the final state s_{end} are known to both the receiver and the transmitter.

[12]Known in a sense that it should be known prior to encoding and decoding.

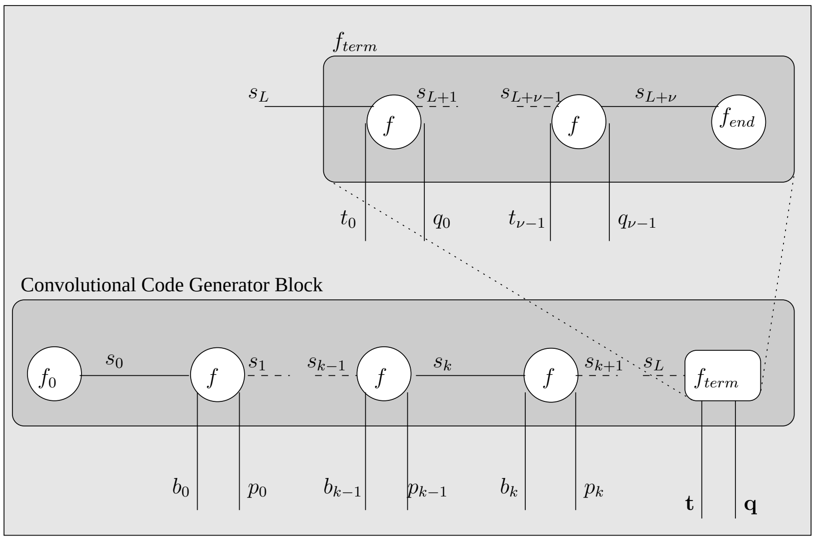


Figure 3.13: *Convolutional Code Generator: factor graph for $K = 1$, $M = 1$ of the indicator function $I\left[[\mathbf{s}, \mathbf{d}] = \chi_1(\mathbf{b})\right]$*

Non-terminated convolutional code We are not required to end up in a known state. The output of the CC generator is denoted by $\mathbf{d} = \left[\mathbf{b}^T \mathbf{p}^T\right]^T$, with state sequence $\mathbf{s} = [s_0, \ldots, s_L]^T$. The initial state $s_0 = s_{start}$ is known to both the receiver and the transmitter.

In either case, we can fully describe the CC generator with the indicator function $I\left[[\mathbf{s}, \mathbf{d}] = \chi_1(\mathbf{b})\right]$, where $\mathbf{b}$ is the binary information sequence, $\mathbf{d}$ contains information and parity bits (and possibly the termination sequences $\mathbf{t}$ and $\mathbf{q}$) and $\mathbf{s}$ is a state sequence. Later, this indicator function will appear in the factorization of the a posteriori distribution $p(\mathbf{b}, \mathbf{d}, \mathbf{s}, \mathbf{x} \,|\mathbf{r})$ where $\mathbf{r}$ is the observation and $\mathbf{x}$ are additional variables (to be defined later).

3.4.4.2 Factor graph

We will now construct a factor graph representing the function $I\left[[\mathbf{s}, \mathbf{d}] = \chi_1(\mathbf{b})\right]$. In order to keep the situation insightful, we'll stick to the case $K = M = 1$ for the remainder of this chapter. In this case, $L = N_b$. For a terminated convolutional code, we can factorize the function as

$$I\left[[\mathbf{s}, \mathbf{d}] = \chi_1(\mathbf{b})\right] = f_0(s_0) \prod_{k=0}^{N_b-1} f(s_k, s_{k+1}, p_k, b_k)\, f_{term}(s_L, \mathbf{t}, \mathbf{q}) \tag{3.26}$$

where, for initial state s_{start} and final state s_{end}:

$$\begin{aligned}
f_0(s_0) &= I\left[s_0 = s_{start}\right] \\
f(s_k, s_{k+1}, p_k, b_k) &= I\left[[p_k,\ s_{k+1}] = f_{all}([b_k,\ s_k])\right] \\
f_{term}(s_L, \mathbf{t}, \mathbf{q}) &= \prod_{n=0}^{\nu-1} I\left[[\mathbf{q}_n,\ s_{N_b+n+1}] = f_{all}([\mathbf{t}_n,\ s_{N_b+n}])\right] f_{end}(s_{L+\nu}) \\
f_{end}(s_{L+\nu}) &= I\left[s_{L+\nu} = s_{end}\right]
\end{aligned}$$

which leads to a factor graph shown in Fig. 3.13. We will refer to this graph as the Convolutional Code Generator Block (CCGB). It has as variables[13] $\mathbf{d}$ and $\mathbf{s}$. This block represents an indicator function which evaluates to $1 \iff \mathbf{d}$ is a valid output with associated state sequence $\mathbf{s}$. The node $f_0(s_0)$ constrains the initial state to be s_{start}. The nodes f enforce the relation (3.25). When the code is not terminated, the node f_{term} from Fig. 3.13 can be omitted. The node $f_{term}(s_L, \mathbf{t}, \mathbf{q})$ is employed in terminated convolutional codes to ensure the final state ($s_{L+\nu}$) equals s_{end}. Breaking open this node, we can reveal its details, as shown in Fig. 3.13.

[13]Note that $\mathbf{b}$ is contained in $\mathbf{d}$.

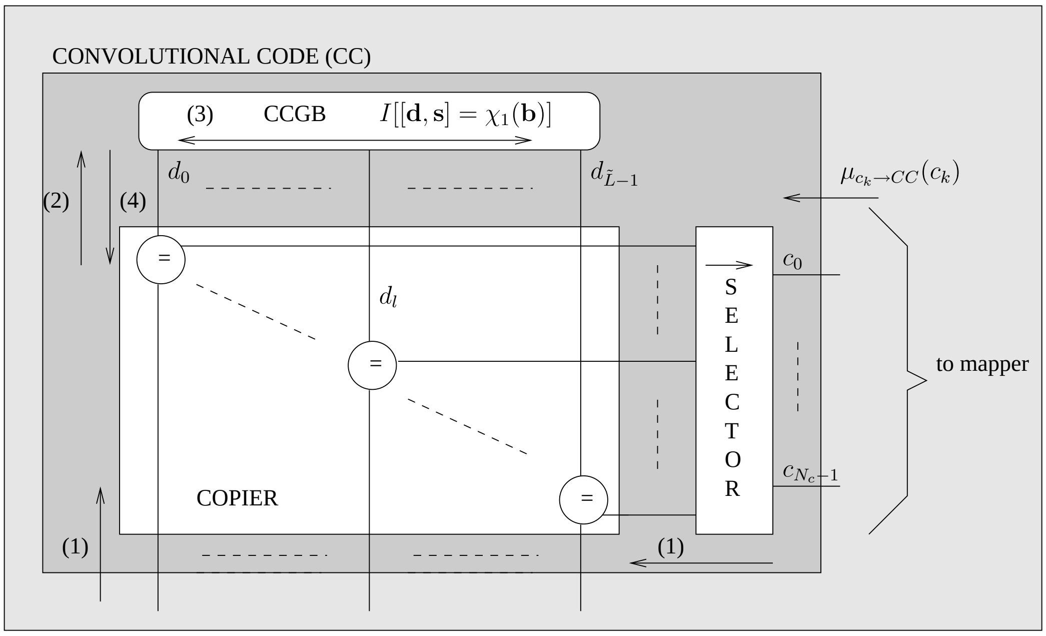


Figure 3.14: *Convolutional code: factor graph for $K = 1$, $M = 1$. $N_c \leq \tilde{L}$. The block marked CCGB is described in Fig. 3.13. The use of the copier will be explained later. The selector punctures a subset of the $\tilde{L}$ bits, retaining N_c bits for transmission over the channel.*

3.4.4.3 Sum-product algorithm

We can apply the sum-product algorithm on the factor graph from Fig. 3.13, assuming messages $\{\mu_{b_k \to CCGB}(b_k)\}$, $\{\mu_{p_k \to CCGB}(p_k)\}$ and, for terminated codes, $\{\mu_{q_k \to CCGB}(q_k)\}$, $\{\mu_{t_k \to CCGB}(t_k)\}$ are available. The application of the sum-product algorithm is fairly straightforward. For a detailed description of the resulting algorithm, the reader is referred to the appendix of this chapter (section 3.6.2).

It turns out that the resulting sum-product algorithm is equivalent to the well-known BCJR algorithm [33], named after its creators Bahl, Cocke, Jelenik and Raviv. The sum-product algorithm has the same computational complexity, but is somewhat more elegant in its mathematical description, since it only requires a single update-rule to describe the entire algorithm.

3.4.5 Convolutional Code

With the building blocks from the previous sections, we can now define a very general convolutional code.

3.4.5.1 Definition

The sequence $\mathbf{d}$ (of size, say, $\tilde{L}$), containing all information bits $\mathbf{b}$, parity bits $\mathbf{p}$ (and, for terminated codes, termination bits $\mathbf{t}$ and $\mathbf{q}$) is not transmitted over the channel directly. We puncture $\tilde{L} - N_c$ bits of $\mathbf{d}$. The remaining N_c bits are then provided to mapping blocks (as described in section 3.4.3), yielding a sequence of constellation symbols $\mathbf{a}$. The rate of this convolutional code is given by N_b/N_c. Note that for a fixed N_b, this rate depends on (i) whether or not the code is terminated, and (ii) on the puncturing pattern.

3.4.5.2 Factor graph

We include two more blocks in the factor graph to come to a view as shown in Fig. 3.14: a *copier* and a *selector*. The selector selects N_c bits at the output of the copier for transmission over the channel; the remaining $\tilde{L} - N_c$ bits are punctured. In the case of a systematic code, the selector will not puncture any of the information bits.

3.4.5.3 Sum-product algorithm

Similar to the generic error-correcting code from section 3.4.2, the convolutional code can be decoded with the sum-product algorithm. Decoding starts from some observation $\mathbf{r}$, with a corresponding observation model. This yields messages $\mu_{c_k \to CC}(c_k)$, $k = 0, \ldots, N_c - 1$, where $[c_0, \ldots, c_{N_c-1}]$ are the coded bits after puncturing. For instance, in the case of BPSK modulation and AWGN with variance σ^2 per real dimension, we have

$$\mu_{c_k \to CC}(c_k) \quad \propto \quad \exp\left(-\frac{1}{2\sigma^2}\left|r_k - (2c_k - 1)\right|^2\right) \tag{3.27}$$

In order to decode the code (i.e., infer b_k from $\mathbf{r}$), we apply the SP algorithm to the graph from Fig. 3.14. Note that the graph has no cycles, so the exact marginals will be computed.

1. **Initialization:** First, messages $\mu_{c_k \to CC}(c_k)$ are sent over the c_k-edges. At the same time, the half-edges in the selector-node transmit messages equal to 1 (this corresponds to the bits that were punctured). Similarly, the half-edges at the bottom transmit upward messages equal to 1.
2. In the copier node, the equal nodes compute an upward message to the CCGB block by multiplying the two incoming messages (according to the SP rule). This yields $\mu_{b_k \to CCGB}(b_k)$ and $\mu_{p_k \to CCGB}(p_k)$, $k = 0, \ldots, L-1$. In the case of a terminated code, this step also yields $\mu_{t_k \to CCGB}(t_k)$ and $\mu_{q_k \to CCGB}(q_k)$, $k = 0, \ldots, \nu - 1$.
3. The SP-algorithm is applied to the CCGB block. This algorithm is described in the appendix of this chapter (section 3.6.2).
4. The CCGB outputs messages $\mu_{CCGB \to b_k}(b_k)$.
5. **Termination:** The APPs of the information bits are then given by

   $$p(b_k \,|\mathbf{r}) \propto \mu_{CCGB \to b_k}(b_k) \times \mu_{b_k \to CCGB}(b_k)$$

   and decisions w.r.t. the information bits can be made:

   $$\hat{b}_k = \arg\max_{b_k} p(b_k \,|\mathbf{r}).$$

Steps 1-4 are marked in Fig. 3.14.

3.4.6 Turbo code

Although turbo codes are a very advanced type of code, their structure and decoding algorithm can be easily described in terms of what we have already covered. Turbo codes are simply two convolutional codes, separated by an interleaver [7]. A possible factor graph is shown in Fig. 3.15 : using a selector, a *subset* of the output bits ($\mathbf{d}$) of the top-most encoder is used, after interleaving, as information bits of a second convolutional encoder. For each encoder, a set of bits is selected for transmission over the channel.

The different selectors (the three selectors S-0, S-1 and S-2 are marked in black in Fig 3.15) in the graph can be modified to change the rate of the code. The interleaver is a crucial part of the code, as it improves the error-correcting abilities of the codes, and increases the length of cycles.

Note that the factor graph of a turbo code always contains cycles. Decoding is generally performed iteratively as described in Algorithm 1. Each of the constituent CC blocks accepts information from the channel, as well as from the other constituent CC. To initialize this process, we have to set initial messages from one decoder to the other, equal to some constant.

3.4.6.1 Parallel and serial concatenation

Two types of turbo codes are generally distinguished: parallel concatenated convolutional codes (PCCC) and serial concatenated convolutional codes (SCCC). The latter are obtained in Fig. 3.15 by

- not transmitting any of the bits of the topmost encoder (i.e., the corresponding selector S-1 punctures all the bits)
- passing *all* the bits from the topmost encoder to the bottom encoder (i.e., the selector S-0 punctures none of the bits).

A PCCC code is obtained by

- passing only the information bits $\mathbf{b}$ from the top encoder to the bottom encoder (i.e., the selector S-0 in Fig. 3.15 punctures all the parity bits and termination bits). Both encoders use the same set of information bits (albeit in interleaved form for the bottom encoder).

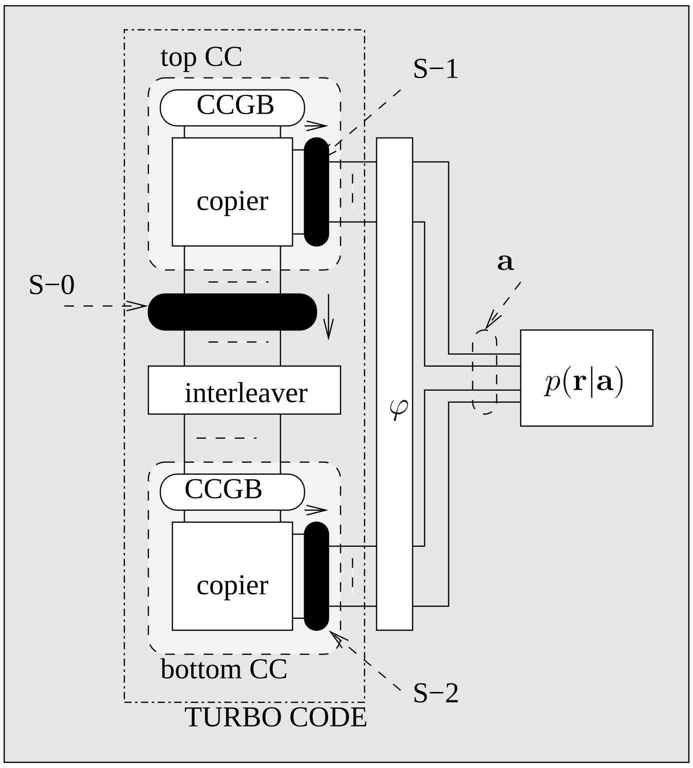


Figure 3.15: *Turbo code, including constituent CC, interleaver, mapper nodes and observation nodes. Three selector blocks perform puncturing and are marked in black. Some additional equality nodes (not depicted) are required to connect the interleaver with the bottom CCGB.*

Algorithm 1 SP algorithm for a turbo code

```
1: Initialization: set messages from bottom CC to top CC to constant value
2: for $i = 1$ to $I_{max}$ do
3:     Decode top CC. Accept messages from channel and from bottom CC.
4:     Decode bottom CC. Accept messages from channel and from top CC.
5: end for
6: compute APPs of information bits
```

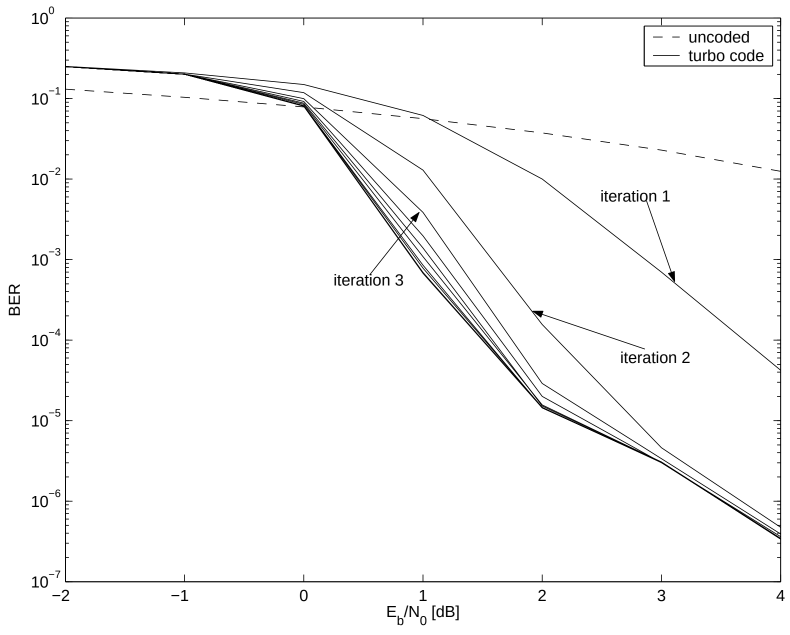


Figure 3.16: *Turbo Code: performance*

3.4.6.2 Illustration

To illustrate the power of turbo-processing, an example of the performance of the iterative SP algorithm for a turbo code is shown in Fig. 3.16. Observe the decreasing bit-error-rate (BER) with each iteration[14].

3.4.7 Further applications

Of course, the story doesn't end there. Many of the most powerful error-correcting codes (such as turbo codes, LDPC codes [6, 34] and RA codes) can be elegantly represented as factor graphs. As we have seen, factor graphs can be combined to make larger factor graphs: for instance, attaching a convolutional code node with an array of mapper nodes. Iterative detection for such systems was originally developed in an ad-hoc way in [35]. Advanced iterative detection schemes for multi-antenna systems are likewise based on factor graphs [36]. Even techniques such as FFTs, the Forward-Backward (Baum-Welch) algorithm on Hidden-Markov models and Kalman filters/smoothers can be interpreted as the application of the sum-product algorithm on factor graphs. A more extensive list of applications of factor graphs in digital communication systems can be found in [8, 37] and references therein.

Nodes in factor graphs can be grouped and made into a 'black box': the black box can then operate according to an algorithm different from the SP algorithm and output messages of the connected variables. For instance, one could replace the CCGB nodes in Fig. 3.13 by another decoding algorithm (such as the Soft Viterbi algorithm [38]). The CCGB accepts the same messages as the standard sum-product algorithm. It outputs messages that can be interpreted as or transformed into valid messages, that in turn can be used elsewhere in the factor graph. However, the messages inside the CCGB will not be computed by the sum-product algorithm, but by the Soft Viterbi algorithm. Although this will affect the performance, the sum-product algorithm can still be applied elsewhere in the graph. Since we will make use of error-correcting codes based on factor graphs throughout this dissertation, many digital components in the receiver are forced to adhere to the input-output relationship described by the sum-product rule.

3.4.8 Some important practical considerations

In most cases, when the factor graph represents a function *proportional* to some a posteriori distribution $p(\mathbf{b}\,|\mathbf{r})$, messages computed during the sum-product algorithm should be *normalized*, so that $\sum_{\alpha} \mu_{f\rightarrow b_k}(b_k = \alpha) = 1$. Since the graph represents $p(\mathbf{b}\,|\mathbf{r})$ up to a multiplicative constant, this normalization does not affect the outcome of the SP algorithm[15]: the APPs of the variables are still obtained after normalization. However, normalizing messages during

[14]Parameters: rate $1/3$ parallel concatenated convolutional code. Constituent codes: systematic recursive, rate $1/2$, generators $(21, 37)_8$. Block size: 501 QPSK symbols. Random interleaving. Only the first convolutional code is terminated.

[15]This also explains the somewhat liberal use of the notation '$\propto$' in the previous sections.

the SP algorithm has two benefits:

- Messages can at all times be interpreted as probability mass functions (or probability density functions, for continuous variables).
- More importantly, normalization avoids problems related to numerical stability: multiplying probabilities tends to create messages with very small magnitude, often beyond the resolution of current processors.

3.5 Main Points

We have presented the concept of factor graphs:

- Factor graphs provide a convenient way to represent the decomposition of functions of many variables;
- applying the sum-product algorithm on a factor graph yields the marginals of the function;
- this marginalization is no longer exact when the graph contains cycles. Short cycles should be avoided;
- The turbo-principle can be interpreted as applying the SP algorithm on a factor graph of a suitable a posteriori distribution $p\left(\mathbf{b}, \mathbf{x}\right| \mathbf{r})$, where $\mathbf{b}$ contains the variables we wish to detect, $\mathbf{r}$ is an observation and $\mathbf{x}$ is a vector of additional variables.

In the field of communications theory, factor graphs have been applied mostly in the context of *adaptation* to known channel and transmit modes and to *data detection*. Recently, factor graphs have begun to appear as a means to *estimate* parameters, with mixed success. One of the main problems with factor graph-based estimation lies in the nature of the parameters that need to be estimated: since some of them are continuous, the messages are no longer probability mass functions[16], but rather probability density functions. These need to be represented in the SP algorithm in an efficient manner.

[16]Probability mass functions can conveniently (and exactly) be described with a vector representation.

3.6 Appendix

3.6.1 The Sum-product algorithm

Theorem 3.6.1 (The Sum-Product algorithm). *Given a function and a cycle-less corresponding factor graph representation, the Sum-Product Algorithm relates outgoing messages to incoming messages, according to*[17]

$$\mu_{f_n \to x_k}(x_k) = \sum_{\sim\{x_k\}} f_n(X_n) \prod_{l \neq k} \mu_{x_l \to f_n}(x_l). \tag{3.28}$$

The SP algorithm starts from the half-edges in the graph and from nodes with degree 1. Each node computes outgoing messages ($\mu_{f_n \to x_k}(x_k)$) when all incoming messages ($\mu_{x_l \to f_n}(x_l)$, $l \neq k$) have been received. Once all messages have been computed, the marginals are then given by (3.7).

Proof. By induction on N, the number of nodes in the graph. A function with N nodes can be written as

$$f^{(N)}(x_k, \mathbf{x}_L, \mathbf{x}_R, x_c) = f_L^{(N)}(x_k, \mathbf{x}_L) \times f_R^{(N)}(x_k, \mathbf{x}_R, x_c) \tag{3.29}$$

where x_k is a variable, $f_L^{(N)}$ ($f_R^{(N)}$) is the part of the graph to the left (right) of x_k and x_c is a variable that will be used during the inductive step. Since the graph is a tree, $\mathbf{x}_L$ and $\mathbf{x}_R$ are distinct variable sets (either of which may be empty). The marginal corresponding to x_k is given by

$$g_k^{(N)}(x_k) = \sum_{\mathbf{x}_L} f_L^{(N)}(x_k, \mathbf{x}_L) \times \sum_{\mathbf{x}_R, x_c} f_R^{(N)}(x_k, \mathbf{x}_R, x_c) \tag{3.30}$$

Messages in a graph with N nodes will be superscripted as $\mu^{(N)}(.)$. We will denote by $\mu_{R2L}^{(N)}(.)$ and $\mu_{L2R}^{(N)}(.)$ the messages on an edge from right-to-left and from left-to-right, respectively. The corresponding factor graph is depicted in Fig. 3.17.

A. Base Case: $N = 1$**.**

This is trivial: the message from the variable to the node is identically 1, while the message from the node to the variable is the marginal.

B. Inductive Hypothesis:

Assume the SP algorithm correctly computes the marginals for any factor graph with up to N nodes.

C. Inductive step:

Start with a cycle-less graph with N nodes. As before, the corresponding function will be denoted $f^{(N)}(\mathbf{x}_L, x_k, \mathbf{x}_R, x_c)$. Without loss of generality, we assume that the new node was attached at the far right of the graph to an edge x_c, so that (see Fig. 3.17)

$$\begin{aligned} & f^{(N+1)}(\mathbf{x}_L, x_k, \mathbf{x}_R, x_c, \mathbf{b}) \\ & = f^{(N)}(\mathbf{x}_L, x_k, \mathbf{x}_R, x_c) \times \psi(x_c, \mathbf{b}) && (3.31) \\ & = f_L^{(N)}(x_k, \mathbf{x}_L) f_R^{(N)}(x_k, \mathbf{x}_R, x_c) \psi(x_c, \mathbf{b}). && (3.32) \end{aligned}$$

We now apply the SP-algorithm on the factor graph of $f^{(N+1)}(\mathbf{x}_L, x_k, \mathbf{x}_R, x_c, \mathbf{b})$. Three types of variables can be discerned: variables in $S_N \doteq \{\mathbf{x}_L, x_k, \mathbf{x}_R\}$, variables in $\{\mathbf{b}\}$ and the variable x_c.

→**Variables in** S_N Let us take the generic element x_k in S_N. The messages from left to right are unchanged as compared to the N-node factor-graph:

$$\begin{aligned} \mu_{L2R}^{(N+1)}(x_k) &= \mu_{L2R}^{(N)}(x_k) && (3.33) \\ &= \sum_{\mathbf{x}_L} f_L^{(N)}(x_k, \mathbf{x}_L). && (3.34) \end{aligned}$$

[17]For continuous variables, the summations are replaced with integrals.

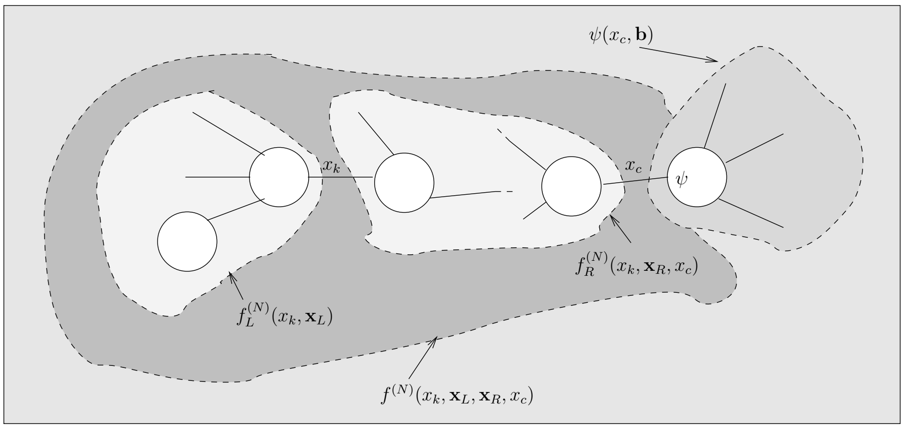

$\psi(x_c,\mathbf{b})$
x_k
x_c
ψ
$f_R^{(N)}(x_k,\mathbf{x}_R,x_c)$
$f_L^{(N)}(x_k,\mathbf{x}_L)$
$f^{(N)}(x_k,\mathbf{x}_L,\mathbf{x}_R,x_c)$

Figure 3.17: *Sum-Product Algorithm on a function with* $N+1$ *factors:* $f_L^{(N)}(x_k,\mathbf{x}_L)\,f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\,\psi(x_c,\mathbf{b})$

On the other hand, the messages from right to left are now different. Since $f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\times\psi(x_c,\mathbf{b})$ contains less than $N+1$ nodes, the SP algorithm yields the following marginal of the function $f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\times\psi(x_c,\mathbf{b})$:

$$\mu_{R2L}^{(N+1)}(x_k) \quad = \quad \sum_{\mathbf{x}_R,\mathbf{b},x_c} f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\times\psi(x_c,\mathbf{b}). \tag{3.35}$$

Multiplying (3.33) and (3.35) yields

$$\mu_{L2R}^{(N+1)}(x_k)\times\mu_{R2L}^{(N+1)}(x_k) \tag{3.36}$$
$$= \sum_{\mathbf{x}_R,\mathbf{x}_L,\mathbf{b},x_c} f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\,f_L^{(N)}(x_k,\mathbf{x}_L)\,\psi(x_c,\mathbf{b}) \tag{3.37}$$
$$= \sum_{\mathbf{x}_R,\mathbf{x}_L,\mathbf{b},x_c} f_R^{(N)}(x_k,\mathbf{x}_R,x_c)\,f_L^{(N)}(x_k,\mathbf{x}_L)\,\psi(x_c,\mathbf{b}) \tag{3.38}$$
$$= \sum_{\mathbf{x}_R,\mathbf{x}_L,\mathbf{b},x_c} f^{(N+1)}(\mathbf{x}_L,x_k,\mathbf{x}_R,x_c,\mathbf{b}) \tag{3.39}$$
$$= g_k^{N+1}(x_k). \tag{3.40}$$

→**Variable** x_c Due to the inductive hypothesis, the message over edge x_c from left to right is given by

$$\mu_{L2R}^{(N+1)}(x_c) \quad = \quad \sum_{\mathbf{x}_L,x_k,\mathbf{x}_R} f_L^{(N)}(x_k,\mathbf{x}_L)\times f_R^{(N)}(x_k,\mathbf{x}_R,x_c). \tag{3.41}$$

On the other hand, since the components in $\mathbf{b}$ correspond to half-edges:

$$\mu_{R2L}^{(N+1)}(x_c) = \sum_{\mathbf{b}}\psi(x_c,\mathbf{b}) \tag{3.42}$$

which immediately yield the correct marginal for x_c.

→**Variables in** $\{\mathbf{b}\}$ Concentrating on a generic element b_l, we know that $\mu_{R2L}^{(N+1)}(b_l)=1$. Taking into account (3.41), we immediately find the correct marginal for b_l.

□

3.6.2 Convolutional Code Generator: sum-product algorithm

Let us consider the generator of a non-terminated CC. Assume, given some observation $\mathbf{r}$, we have computed the messages $\mu_{b_k\to CCGB}(b_k)$ and $\mu_{p_k\to CCGB}(p_k)$ for $k=0,\ldots,L-1$. We are interested in determining the marginal APPs of the information bits b_k. We introduce the following notations (see Fig. 3.18):

- $\mu_{b_k \to CCGB}(b_k) \doteq \mu_{b_k}^{(U)}(b_k)$ and $\mu_{p_k \to CCGB}(p_k) \doteq \mu_{p_k}^{(U)}(p_k)$, with 'U' for upward message.
- messages related to state variables will be denoted $\mu_{s_k}^{(L)}(s_k)$ and $\mu_{s_k}^{(R)}(s_k)$ for Left-ward and Right-ward messages, respectively.

We know $\mu_{s_0}^{(R)}(s_0)$ since the code starts from a known state. On the other hand, we know that $\mu_{s_L}^{(L)}(s_L)$ is a constant, since the code is not terminated so that all final states are equiprobable. We can now apply the sum-product algorithm on the graph from Fig. 3.13. This immediately yields Algorithm 2. We remind that a node can compute an outgoing message, only when all incoming messages are available. The algorithm starts with two passes to compute state-messages: a forward pass and a backward pass, which are executed in parallel. This is followed by the computation of downward messages of information and parity bits. How the messages are related, is depicted in Fig. 3.18.

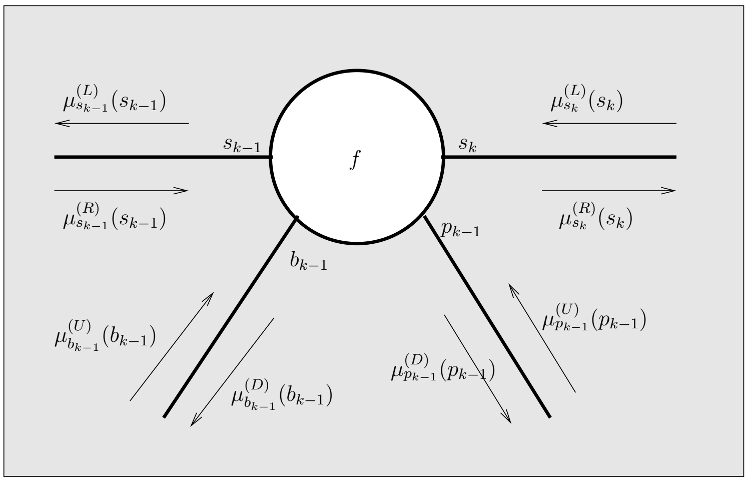
Algorithm 2 SP algorithm for convolutional code

```
1: Define: $\mathbf{x}_k = [s_{k-1}, s_k, p_{k-1}, b_{k-1}], \forall k$
2: for $k = 1$ to $L$ do
3:     Forward pass (Left to Right)
           $$\mu_{s_k}^{(R)}(s_k) = \sum_{\mathbf{x}_k : s_k} f(\mathbf{x}_k)\, \mu_{b_{k-1}}^{(U)}(b_{k-1})\, \mu_{p_{k-1}}^{(U)}(p_{k-1})\, \mu_{s_{k-1}}^{(R)}(s_{k-1})$$
4:     Backward pass (Right to Left)
           $$\mu_{s_{L-k}}^{(L)}(s_{L-k}) = \sum_{\mathbf{x}_{L-k+1} : s_{L-k}} f(\mathbf{x}_{L-k+1})\, \mu_{b_{L-k}}^{(U)}(b_{L-k})\, \mu_{p_{L-k}}^{(U)}(p_{L-k})\, \mu_{s_{L-k+1}}^{(L)}(s_{L-k+1})$$
5: end for
6: for $k = 0$ to $L - 1$ do
7:     Downward messages for information bits
           $$\mu_{b_k}^{(D)}(b_k) = \sum_{\mathbf{x}_{k+1} : b_k} f(\mathbf{x}_{k+1})\, \mu_{s_{k+1}}^{(L)}(s_{k+1})\, \mu_{s_k}^{(R)}(s_k)\, \mu_{p_k}^{(U)}(p_k)$$
8:     Downward messages for parity bits
           $$\mu_{p_k}^{(D)}(p_k) = \sum_{\mathbf{x}_{k+1} : p_k} f(\mathbf{x}_{k+1})\, \mu_{s_{k+1}}^{(L)}(s_{k+1})\, \mu_{s_k}^{(R)}(s_k)\, \mu_{b_k}^{(U)}(b_k)$$
9: end for
```

Once the upward messages and downward messages are known, the marginal APPs are given by

$$p(b_k | \mathbf{r}) \propto \mu_{b_k}^{(U)}(b_k) \times \mu_{b_k}^{(D)}(b_k).$$

The presented algorithm is equivalent to the well-known BCJR algorithm [33], named after its creators Bahl, Cocke, Jelenik and Raviv. Applying the sum-product algorithm has the same computational complexity, but is somewhat more elegant in its mathematical description, since it only requires a single update-rule to describe the entire algorithm.



$\mu^{(L)}_{s_{k-1}}(s_{k-1})$
$\mu^{(L)}_{s_k}(s_k)$
s_{k-1}
f
s_k
$\mu^{(R)}_{s_{k-1}}(s_{k-1})$
$\mu^{(R)}_{s_k}(s_k)$
p_{k-1}
b_{k-1}
$\mu^{(U)}_{b_{k-1}}(b_{k-1})$
$\mu^{(U)}_{p_{k-1}}(p_{k-1})$
$\mu^{(D)}_{p_{k-1}}(p_{k-1})$
$\mu^{(D)}_{b_{k-1}}(b_{k-1})$

Figure 3.18: *Sum-product algorithm on convolutional code*

Part II

Multi-mode receivers: adaptation

Outline

In the second part of this dissertation, we will investigate how a receiver can detect the transmitted data in the presence of *known* channel parameters (including synchronization parameters) and a *known* transmit mode. We show how the receiver could be modified when the channel parameters and/or the transmit mode change, and how such modifications can be accomplished efficiently.

- We start this part with a brief overview of some basic concepts related to continuous-time and discrete-time signals (Chapter 4). We include some basic notions related to sampling and digital filtering, as well as the semi-analytical computation of BER degradations.
- In Chapter 5 we consider problems of timing correction, equalization and data detection. We show how the receiver needs to be modified to operate when the channel state is changed. With the aid of factor graphs, iterative receivers are derived that will be useful later on.
- The problem of mode adaptation (i.e., how to modify the receiver as the transmit mode changes) is covered in Chapter 6. We show that in most cases adaptation is a fairly straightforward task and boils down to replacing parts in factor graphs to suit the current transmit mode. An important exception is adaptation to the symbol rate in multi-rate transmission schemes.
- The design of receivers that are able to operate efficiently for multi-rate transmission is the main topic of Chapter 7. Various receiver structures will be discussed and compared in terms of their BER degradation, as compared to an 'optimal' reference receiver. We propose a low-complexity multi-rate receiver, combining techniques from IF-sampling and sample rate conversion based on a frequency-domain view.

Chapter 4

Basic Principles: Signals and Filters

4.1 Introduction

In this chapter we will cover a mish-mash of aspects related to digital and analog signal processing, including filter theory, baseband and bandpass sampling theorems, Fourier transforms, signal reconstruction, anti-aliasing filters, BER degradation computation and vector-representations of signals. These topics will be applied mainly in Chapter 7. Readers familiar with these topics can safely skip this chapter.

4.2 Continuous-time signals

4.2.1 Signal representation

A signal $x(t)$ and its Fourier Transform (FT) $X(f)$ are related by

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-j2\pi ft} dt \tag{4.1}$$

$$x(t) = \int_{-\infty}^{+\infty} X(f)\, e^{j2\pi ft} df. \tag{4.2}$$

When $X(f) = 0$ for $|f| > B$ we say that $x(t)$ is band-limited to $\pm B$ with a one-sided bandwidth B. When $X(f) = 0$ for $||f| - f_0| > B$, with $f_0 > B$ we say that $x(t)$ is band-limited to

$$f \in [-f_0 - B, -f_0 + B] \cup [f_0 - B, f_0 + B] \tag{4.3}$$

with a one-sided bandwidth $2B$.

4.2.2 Filtering

When we filter $x(t)$ with a linear, time-invariant filter $h(t)$, the resulting signal is the convolution of $x(t)$ and $h(t)$:

$$y(t) = \int_{-\infty}^{+\infty} x(u)\, h(t-u)\, du. \tag{4.4}$$

In the frequency-domain this becomes

$$Y(f) = X(f)\, H(f). \tag{4.5}$$

4.3 Discrete-time signals

4.3.1 Signal representation

When we sample a signal $x(t)$ at a rate $1/T_s$, the Discrete-Time Fourier Transform (DTFT) is given by

$$X\left(e^{j2\pi fT_s}\right) = \sum_{k=-\infty}^{+\infty} x(kT_s)\, e^{-j2\pi fkT_s} \tag{4.6}$$

$$= \frac{1}{T_s}\sum_{k=-\infty}^{+\infty} X\left(f-\frac{k}{T_s}\right) \tag{4.7}$$

Observe that $X\left(e^{j2\pi fT_s}\right)$ is periodic, with period $1/T_s$. When $X\left(f-\frac{k}{T_s}\right)$ and $X\left(f-\frac{k'}{T_s}\right)$ $(k \neq k')$ overlap, *aliasing* occurs, and the original signal $x(t)$ can no longer be reconstructed. This idea is formalized in the following two sampling theorems.

Theorem 4.3.1 (The Baseband Sampling Theorem). *A signal that is band-limited to the frequency band $f \in [-B, +B]$ can be fully reconstructed from samples taken at a rate $1/T_s$, when*

$$\frac{1}{T_s} \geq 2B. \tag{4.8}$$

Proof. Due to the periodicity, it suffices to verify that $X(f)$ and $X(f-1/T_s)$ do not overlap. $X(f)$ contains no component for $f > B$, while $X(f-1/T_s)$ contains no components for $f < 1/T_s - B$. Since $B < 1/T_s - B$, there is no aliasing. Hence, the signal can be reconstructed from its samples.

□

Theorem 4.3.2 (The Bandpass Sampling Theorem). *A signal that is band-limited to the frequency band $f \in [-f_0 - B, -f_0 + B] \bigcup [f$ can be reconstructed from samples, taken at a rate $1/T_s$, when*

$$\frac{1}{T_s} \geq \frac{2B}{\min(r, 1-r)} \tag{4.9}$$

where r is the fractional part of $2f_0T_s$ (i.e., $r = rem(2f_0T_s) \in [0, 1[$).

Proof. We denote the signal component centered at $f = f_0$ (resp. $f = -f_0$) by $X^+(f)$ (resp. $X^-(f)$). Let us focus on $X^+(f)$. Since $1/T_s \geq 2B$, the baseband sampling theorem tells us that the periodic extension of $X^+(f)$ will not overlap with $X^+(f)$. The same is true for $X^-(f)$. Hence, it suffices to show that the periodic extension of $X^-(f)$ does not overlap with $X^+(f)$. The components of the periodic extension of $X^-(f)$ closest to $X^+(f)$ are at $f = -f_0 + k_1/T_s$ and $f = -f_0 + k_2/T_s$ with $k_1 = \lfloor 2f_0T_s \rfloor$ and $k_2 = \lceil 2f_0T_s \rceil = k_1 + 1$. This situation is depicted in Fig. 4.1, where r is defined as

$$r = 2f_0T_s - \lfloor 2f_0T_s \rfloor \tag{4.10}$$

$$= rem(2f_0T_s) \tag{4.11}$$

There is no aliasing when

$$r/T_s > 2B \tag{4.12}$$

and

$$(1-r)/T_s > 2B. \tag{4.13}$$

□

4.3.2 Filtering

When we filter $x(kT_s)$ with a digital linear, time-invariant filter $h(kT_s)$, (which can be interpreted as a sampled continuous-time filter), the output is given by

$$y_d(kT_s) = T_s \sum_{l=-\infty}^{+\infty} x(lT_s)\, h(kT_s - lT_s) \tag{4.14}$$

In the frequency-domain this becomes

$$Y_d\left(e^{j2\pi fT_s}\right) = X\left(e^{j2\pi fT_s}\right) \tilde{H}\left(e^{j2\pi fT_s}\right) \tag{4.15}$$

where $\tilde{H}\left(e^{j2\pi fT_s}\right) = T_s H\left(e^{j2\pi fT_s}\right)$.

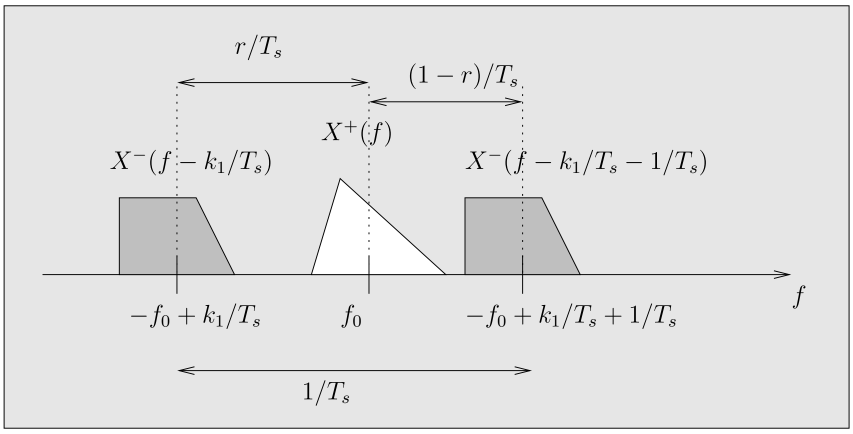


Figure 4.1: *Bandpass sampling theorem*

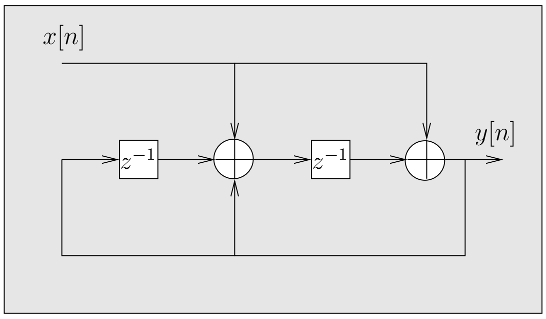


Figure 4.2: *An IIR filter*

Digital and Analog Processing

When processing an analog signal $x\left(t\right)$ with an analog filter $h\left(t\right)$, suppose that $x\left(t\right)$, $h\left(t\right)$ and $1/T_s$ satisfy the following conditions

1. both $x\left(t\right)$ and $h\left(t\right)$ are band-limited to $\pm B$
2. the sample rate $1/T_s$ satisfies the baseband sampling theorem (condition (4.8))
3. the summation in (4.14) goes from the infinite past to the infinite future.

Then, we have the following relationship between the continuous-time signal $y\left(t\right)$ from (4.4) and its discrete-time counterpart $y_d\left(kT_s\right)$ from (4.14):

$$y\left(t\right)\big|_{t=kT_s} = y_d\left(kT_s\right). \tag{4.16}$$

In practice, condition (1) and (3) can only be met approximately.

Representation

Digital filters come in two flavors: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. Their DTFTs are commonly written as a function of $z^{-1} = e^{j2\pi fT_s}$:

$$Y_d\left(z\right) = X\left(z\right)H\left(z\right). \tag{4.17}$$

For FIR filters, $H\left(z\right)$ is a finite polynomial in z^{-1}. For IIR filters, $H\left(z\right)$ is the ratio of two finite polynomials in z^{-1}. In Fig. 4.2 we show a simple example for $H\left(z\right) = \left(1+z^{-1}\right)/\left(1+z^{-1}+z^{-2}\right)$. IIR filters allow for more flexible designs, but are potentially unstable.

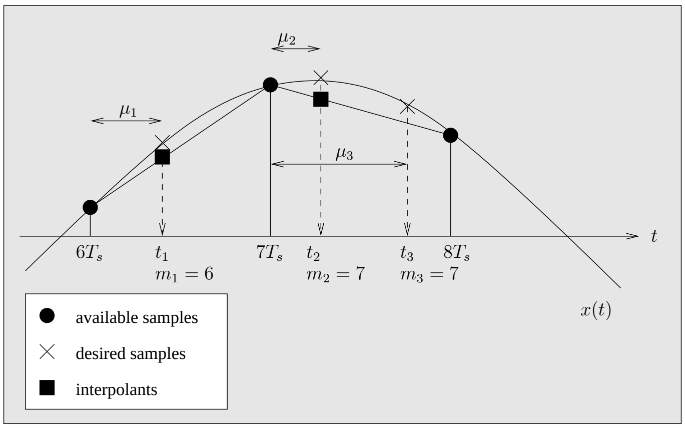


Figure 4.3: *Interpolator: time domain view*

4.4 Two important digital filters

4.4.1 Interpolation filters

Time-domain interpretation

Interpolation is a technique to reconstruct, from samples $x(nT_s)$, a signal $x(t)$ at time instants t_k, according to

$$x_I(t_k) = \sum_{n=-\infty}^{+\infty} x(nT_s)\, p_I(t_k - nT_s) \tag{4.18}$$

where $p_I(t)$ is the *interpolating pulse*, and $x_I(t_k)$ is referred to as an interpolant. Denoting the FT of $x(t)$ by $X(f)$, interpolation is ideal (i.e., $x_I(t_k) = x(t_k)$, $\forall t_k$) provided $X(f) = 0$ for $|f| > 1/(2T_s)$ and $p_I(t) = sinc(t/T_s)$, with $sinc(x) = \sin(\pi x)/(\pi x)$.

We introduce the quantities $m_k \in \mathbb{Z}$ and $\mu_k \in [0, 1[$, uniquely defined by $t_k = m_k T_s + \mu_k T_s$. Hence $m_k T_s$ is the sampling instant of the fixed master clock immediately before or at the instant t_k. Then (4.18) can be transformed into [39,40]

$$x_I(t_k) = \sum_{i=-\infty}^{+\infty} h_i(\mu_k)\, x(m_k T_s - iT_s) \tag{4.19}$$

$$\approx \sum_{i=-N_1}^{N_2} h_i(\mu_k)\, x(m_k T_s - iT_s) \tag{4.20}$$

where $h_i(\mu_k) = p_I(iT_s + \mu_k T_s)$ and N_1 and N_2 can be chosen by the interpolator designer. This indicates that the interpolator can be implemented as a *time-varying* discrete-time filter with coefficients $h_i(\mu_k)$. We see that in a practical implementation, the interpolator has a finite length. For example, for a linear interpolator, we have $p_I(t) = 1 - |t|/T_s$ for $|t| < T_s$ and $p_I(t) = 0$ for $|t| > T_s$; this yields only two nonzero filter taps, i.e., $h_{-1}(\mu) = \mu$ and $h_0(\mu) = 1 - \mu$. Such polynomial interpolators may be implemented in an efficient Farrow structure [41]. The finite number of filter taps inevitably gives rise to non-ideal interpolation (i.e., $x_I(t_k) \neq x(t_k)$). An example is shown in Fig. 4.3.

In many applications, $t_k = kT_I + \tau$ for some output sampling rate $1/T_I$ and some delay τ.

Frequency-domain interpretation

The interpolator output samples can be interpreted as passing the signal

$$\tilde{x}(t) = \sum_{k=-\infty}^{+\infty} x(kT_s)\, \delta(t - kT_s) \tag{4.21}$$

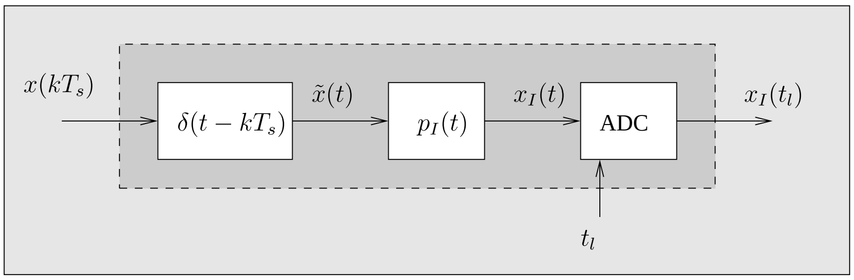


Figure 4.4: *Interpolator: equivalent representation*

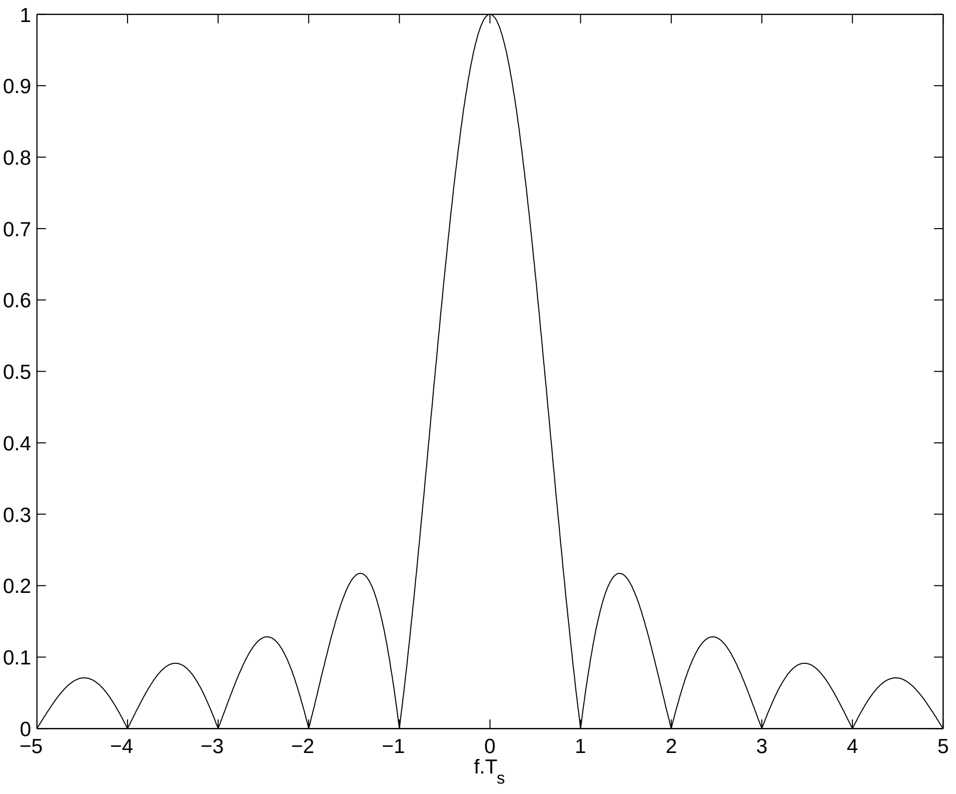


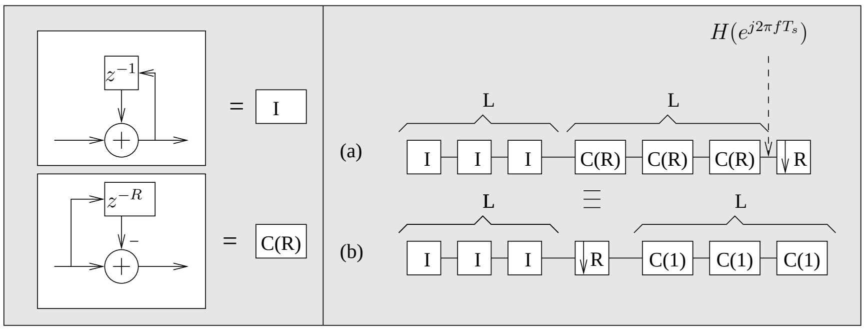
Figure 4.5: *Linear interpolator: Fourier transform $P_I(f)$*

through an interpolator filter with impulse response $p_I(t)$ and re-sampling the output (i.e., the continuous-time reconstructed signal) at time instants $t_l = lT_I + \tau$. This equivalent interpretation is depicted in Fig. 4.4.

The Fourier transform of the interpolator input signal is given by $\tilde{X}(f) = X\left(e^{j2\pi f T_s}\right)$. The interpolator has a FT $P_I(f)$. The interpolator output signal, $x_I(t)$ has a FT $X_I(f)$, given by

$$X_I(f) = X\left(e^{j2\pi f T_s}\right) P_I(f). \tag{4.22}$$

The goal of the interpolator is to remove potential aliasing components, while at the same time not to distort the useful signal too much. Hence, polynomial interpolators have a frequency response which is fairly flat around $f = 0$ and has broad nulls around non-zero multiples of k/T_s, $k \in \mathbb{Z}$. This property is illustrated in Fig. 4.5 for a linear interpolator. Resampling at times $lT_I + \tau$ yields a DTFT equal to



z^{-1}
= I
z^{-R}
$-$
= C(R)
$H(e^{j2\pi fT_s})$
L
L
(a)
I I I C(R) C(R) C(R) $\downarrow$R
$\equiv$
L
L
(b)
I I I $\downarrow$R C(1) C(1) C(1)

Figure 4.6: *Decimating CIC filter of order L with decimation factor R*

$$X_I\left(e^{j2\pi fT_I}\right) \tag{4.23}$$
$$= \frac{1}{T_I}\sum_{m=-\infty}^{+\infty} X_I\left(f-\frac{m}{T_I}\right)e^{-j2\pi(f-m/T_I)\tau} \tag{4.24}$$
$$= \frac{1}{T_IT_s}\sum_{m,n=-\infty}^{+\infty} P_I\left(f-\frac{m}{T_I}\right)X\left(f-\frac{m}{T_I}-\frac{n}{T_s}\right)e^{-j2\pi(f-m/T_I)\tau} \tag{4.25}$$
$$= \frac{1}{T_IT_s}\sum_{m,n=-\infty}^{+\infty} Q_n\left(f-\frac{m}{T_I}\right) \tag{4.26}$$

where

$$Q_n(f) = X\left(f-\frac{n}{T_s}\right)P_I(f)\,e^{-j2\pi f\tau}. \tag{4.27}$$

Generally interpolators are constrained so that $x_I(mT_s) = x(mT_s)$. Hence,

$$\frac{1}{T_s}\sum_{l=-\infty}^{+\infty} P_I\left(f-\frac{l}{T_s}\right) = 1. \tag{4.28}$$

Potential problems

While interpolators are good at reconstructing signals, they are very sensitive to interfering signals. The interpolator removes signal components at k/T_s, $k \neq 0$. However, an interfering signal in the bandwidth $[-1/(2T_s), +1/(2T_s)]$ is attenuated but not fully suppressed. Such a signal may cause aliasing at the output of the interpolator. To see this consider the following example: assume we have a perfect interpolator, but there is an interfering signal centered at $f = 1/(2T_s)$. Further, assume the interpolator output rate is $1/(2T_s)$. Clearly, at the output of the interpolator the useful signal and the interfering signal will be centered around (multiples of) $f = 1/(2T_s)$, so that the useful signal is irrevocably destroyed.

4.4.2 CIC filters

Decimating CIC (Cascaded Integrator Comb) filters are low-complexity decimation filters with good anti-aliasing properties [42]. They are generally used in form (b) in Fig. 4.6: a CIC decimation filter of order L with decimation factor R ($R, L \in \mathbb{N}$) consists of L integrators followed by an order R decimator and L first order differentiators. A mathematically equivalent (but computationally more demanding) form is shown as (a) in Fig. 4.6. The magnitude response can be shown to be (up to an irrelevant constant):

$$\left|H\left(e^{j2\pi fT_s}\right)\right| \propto \left|\frac{\sin(\pi fRT_s)}{\sin(\pi fT_s)}\right|^L. \tag{4.29}$$

This transfer function should be interpreted as the transfer function before the decimator in form (a) in Fig. 4.6. Note that the integrators are IIR filters, and the differentiators are FIR filters.

An example is depicted in Fig. 4.7 for $R = 7$. Observe the nulls at $fT_s = k/R$, $k \neq 0$.

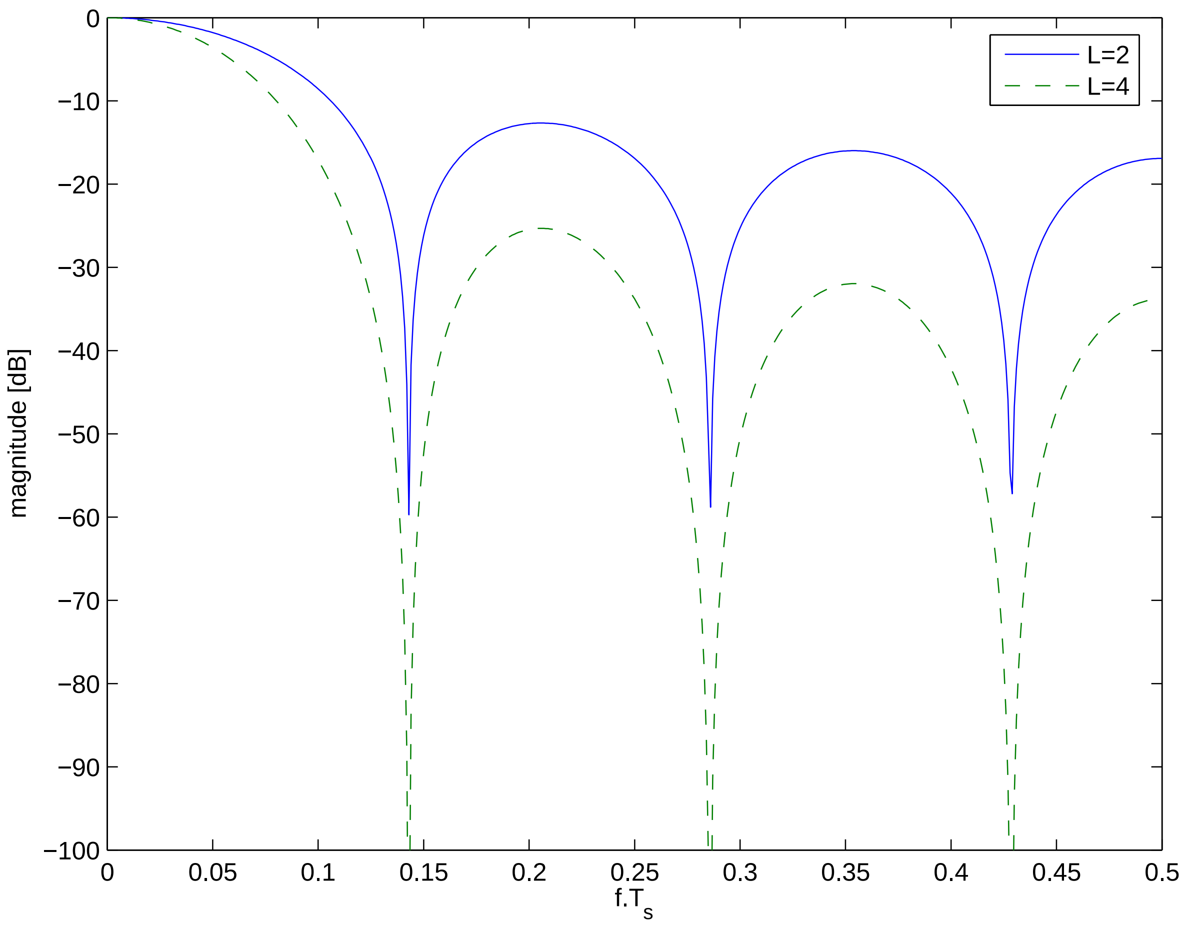


Figure 4.7: *Decimating CIC filter: magnitude frequency response with $R = 7$ for $L = 4$ and $L = 2$.*

Hence, signal components that may cause aliasing after decimation will be removed thanks to the nulls of the filter. As the filter order (L) increases, these nulls become more pronounced. However, this comes at a cost of an increased passband 'droop' (i.e., more attenuation of the useful signal component around $fT_s = 0$).

Recently, CIC filters have been gaining a lot of attention from the research community in the context of Software Radio: their low complexity makes them attractive for a number of different tasks [43].

4.5 BER degradation

In the previous section, we considered two type of filters that distort the signal, in a sense that the input signal can never be reconstructed from the output signal. It is important to evaluate to what extent such an operation degrades the performance of the system. In principle, this can be achieved by implementing the system, performing simulations and inspecting the results. As this is generally time-consuming, we will consider a different approach. Using a (semi-)analytical technique, we will determine the BER degradation of a receiver, compared to a reference receiver. This technique is based on [16, 44, 45].

4.5.1 Principle

Consider the following problem. We transmit a number of symbols $\{a_k\}$. At the receiver, we have an equal number of samples $\{y_k\}$ and a decision on symbol a_k is made, based solely on sample y_k, with:

$$y_k = \sqrt{E_s} a_k s_0(\mathbf{u}) + \sqrt{E_s} \sum_{n \neq k} a_n s_{k-n}(\mathbf{u}) + \sqrt{N_0/2}\,\sigma(\mathbf{u})\, n_k \tag{4.30}$$

where E_s is the energy per symbol, $N_0/2$ is the noise variance per real dimension, $s_0(\mathbf{u})$ is the complex gain of the useful signal component, $s_{k-n}(\mathbf{u})$ is the complex gain of the n-th symbol a_n. The samples n_k are $\mathcal{CN}(0,2)$, while $\sigma(\mathbf{u})$ is a noise scaling factor. The vector $\mathbf{u}$ denotes a set of nuisance parameters, whose values may or may not be known to the receiver. For instance, a timing error, a phase error, non-ideal interpolation, filtering, etc.

The receiver operates under the (incorrect) assumption that $s_n(\mathbf{u}) = \delta_n$ and $\sigma(\mathbf{u}) = 1$. Our goal is to determine the BER degradation of this receiver, compared to a receiver that operates on the samples $\sqrt{E_s} a_k + \sqrt{N_0/2} n_k$. The latter receiver will be named the 'reference receiver'.

The BER degradation (expressed in decibels) of our receiver compared to the reference receiver at a target BER (say BER_{ref}) is defined as

$$\text{BERdeg} = 10 \log_{10} \left(\frac{E_s/N_0}{(E_s/N_0)_{ref}} \right) \tag{4.31}$$

where $(E_s/N_0)_{ref}$ is the SNR required for the reference receiver to attain a BER equal to BER_{ref}, while E_s/N_0 is the SNR required for our receiver to attain the same BER. Denoting with $\mathbf{a}$ all symbols except a_k, and defining

$$f(\mathbf{u}, \mathbf{a}) = \frac{s_0(\mathbf{u}) + \sum_{n \neq k} a_n s_{k-n}(\mathbf{u})}{\sigma(\mathbf{u})} \tag{4.32}$$

we show in the Appendix of this chapter (section 4.8) that the BER degradation can be approximated as

$$\text{BERdeg} = -10 \log_{10} \left(A^2 - V 2 (E_s/N_0)_{ref} \right) \tag{4.33}$$

where

$$A = E_{\mathbf{a},\mathbf{u}} [f(\mathbf{u}, \mathbf{a})] \tag{4.34}$$

$$V = Var_{\mathbf{a},\mathbf{u}} [f(\mathbf{u}, \mathbf{a})] . \tag{4.35}$$

The approximation (4.33) is valid for $2 (E_s/N_0)_{ref} V \ll 1$ and for small degradations (less than 1 dB).

4.5.2 Example

Suppose we transmit a sequence of real data symbols $\{a_k\}$. The receiver operates under an *unknown timing error* τ, with some a priori distribution $p(\tau)$. The observations at the receiver can be shown to be

$$y_k = \sum_{n=-\infty}^{+\infty} a_{k-n} q(nT - \tau) + n_k$$

where $n_k \sim \mathcal{N}\left(0, \frac{1}{2} (N_0/E_s)_{ref}\right)$ and $q(t)$ is a Nyquist pulse, so that $q(kT) = \delta_k$. Hence, $\mathbf{u} = [\tau]$, and $f(\mathbf{u}, \mathbf{a}) = \left(q(-\tau) + \sum_{n \neq 0} a_{k-n} q(nT - \tau)\right) / \sigma(\mathbf{u})$. This yields

$$\begin{aligned} \sigma(\mathbf{u}) &= 1 \\ A &= E_\tau [q(-\tau)] \\ V &= E_\tau \left[\sum_{n=-\infty}^{+\infty} q^2(nT - \tau) \right] - A^2 \end{aligned}$$

This can be further calculated by numerical evaluation of the expectation w.r.t. τ. Note that, when $p(\tau) = \delta(\tau)$, $A = 1$ and $V = 1 - A^2 = 0$, so that

$$\begin{aligned} \text{BERdeg} &= -10 \log_{10}(1) \\ &= 0\,\text{dB}. \end{aligned}$$

Alternatively, we can use a frequency-domain interpretation, based on the characteristic function of τ:

$$\phi(\omega) = \int_{-\infty}^{+\infty} p(\tau)\, e^{-j\tau\omega} d\tau.$$

Since

$$q(nT-\tau) = \int_{-\infty}^{+\infty} Q(f)\, e^{j2\pi f(nT-\tau)} df$$

one easily finds

$$A = \int_{-\infty}^{+\infty} Q(f)\, \phi(2\pi f)\, df$$

$$V = \sum_{n=\infty}^{+\infty} \phi(-2\pi n/T)\, E_n - A^2$$

where $E_n = \int_{-\infty}^{+\infty} Q(f)\, Q(f-n/T)\, df$.

4.6 Signal Representation

4.6.1 Principle

In many places in this dissertation, we will consider vector representations of signals. In general, a vector representation $\mathbf{r}$ of a signal $r(t)$ is obtained by expanding $r(t)$ onto a set of orthonormal basis functions: $\{\phi_0(t), \phi_1(t), \ldots, \phi_{N-1}(t)\}$ as follows [46]:

$$r_k = \int_{-\infty}^{+\infty} r(t)\, \phi_k(t)\, dt \tag{4.36}$$

so that $\mathbf{r} = [r_0, r_1, \ldots, r_{N-1}]^T$. Since the basis functions are orthonormal, we have

$$r(t) = \sum_{k=0}^{N-1} r_k \phi_k(t). \tag{4.37}$$

Sampling a signal then corresponds to a particular set of basis functions (i.e., delay-shifted sinc-pulses). In many cases $N = +\infty$. When the signal is a random process, the values $\{r_k\}$ will be random variables.

4.6.2 Application

A situation that will appear frequently throughout this text is the computation of likelihood functions. Suppose we have a signal $s(t)$ and observe $r(t)$ with:

$$r(t) = s(t) + n(t) \tag{4.38}$$

where $n(t)$ is a complex white Gaussian noise process with PSD $N_0/2$ per real dimension. Projection onto an orthonormal basis yields

$$\mathbf{r} = \mathbf{s} + \mathbf{n} \tag{4.39}$$

where

$$p(\mathbf{n}) \propto \exp\left(-\frac{1}{N_0}\|\mathbf{n}\|^2\right) \tag{4.40}$$

$$\propto \exp\left(-\frac{1}{N_0}\int_{-\infty}^{+\infty} |n(t)|^2\, dt\right) \tag{4.41}$$

The likelihood function $p(\mathbf{r}|\,\mathbf{s})$ can be written in two ways:

$$p(\mathbf{r}|\,\mathbf{s}) \propto \exp\left(-\frac{1}{N_0}\|\mathbf{r}-\mathbf{s}\|^2\right) \tag{4.42}$$

or as

$$p(\mathbf{r}|\,\mathbf{s}) \propto \exp\left(-\frac{1}{N_0}\int_{-\infty}^{+\infty} |r(t)-s(t)|^2\, dt\right). \tag{4.43}$$

4.7 Main points

In this chapter, we discussed the representation of continuous-time and discrete-time signals and filters. Particular attention was paid to interpolation filters and CIC filters. We also described a technique that enables us to directly calculate BER degradations without the need to resort to time-consuming simulations.

4.8 Appendix: BER degradation

4.8.1 Definitions

The BER degradation of our receiver compared to the reference receiver at a target BER (say BER_{ref}) is defined as

$$\text{BERdeg} = 10\log\left(\frac{E_s/N_0}{(E_s/N_0)_{ref}}\right) \tag{4.44}$$

where $(E_s/N_0)_{ref}$ is the SNR required for the reference receiver to attain a BER equal to BER_{ref}, while E_s/N_0 is the SNR required for our receiver to attain the same BER. For BPSK transmission, the BER of the reference receiver is given by

$$BER_{ref}\left((E_s/N_0)_{ref}\right) = Q\left(\sqrt{2\left(\frac{E_s}{N_0}\right)_{ref}}\right) \tag{4.45}$$

with

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{+\infty} e^{-\frac{t^2}{2}}dt. \tag{4.46}$$

The decision variable is given by

$$y_k = \sqrt{E_s}a_k s_0(\mathbf{u}) + \sqrt{E_s}\sum_{n\neq k} a_n s_{k-n}(\mathbf{u}) + \sqrt{N_0/2}\sigma(\mathbf{u})\, n_k. \tag{4.47}$$

For our receiver, we have the following BER vs SNR relationship:

$$BER(E_s/N_0) = E_{\mathbf{a},\mathbf{u}}\left[Q\left(\sqrt{f^2(\mathbf{u},\mathbf{a})\frac{2E_s}{N_0}}\right)\right] \tag{4.48}$$

where

$$\begin{aligned} f(\mathbf{u},\mathbf{a}) &= \frac{s_0(\mathbf{u}) + \sum_{n\neq k} a_n s_{k-n}(\mathbf{u})}{\sigma(\mathbf{u})} && (4.49)\\ &\doteq \frac{f_N(\mathbf{u},\mathbf{a})}{\sigma(\mathbf{u})} && (4.50)\end{aligned}$$

and averaging in (4.48) is performed w.r.t. $\mathbf{u}$, noise statistics and all symbols $\mathbf{a}$, except symbol a_k. Since the reference receiver and our receiver should attain the same BER, this yields:

$$Q\left(\sqrt{2\left(\frac{E_s}{N_0}\right)_{ref}}\right) = E_{\mathbf{a},\mathbf{u}}\left[Q\left(\sqrt{f^2(\mathbf{u},\mathbf{a})\frac{2E_s}{N_0}}\right)\right]. \tag{4.51}$$

For a given value of $(E_s/N_0)_{ref}$, E_s/N_0 can be found iteratively, through a computationally intensive search.

4.8.2 Semi-analytical approach

Searching E_s/N_0 directly can be avoided as follows [44, 45]. We introduce the following notations $e_0 \doteq (E_s/N_0)_{ref}$, $e \doteq E_s/N_0$, $P(e_0) \doteq BER_{ref}\left((E_s/N_0)_{ref}\right)$, $BER(e) \doteq E_{\mathbf{a},\mathbf{u}}\left[P\left(f^2(\mathbf{u},\mathbf{a})\,e\right)\right]$, $A \doteq E_{\mathbf{a},\mathbf{u}}[f(\mathbf{u},\mathbf{a})]$, and $V \doteq E_{\mathbf{a},\mathbf{u}}\left[f^2(\mathbf{u},\mathbf{a})\right] - A^2$. Similar to [44, 45], we model $f(\mathbf{u},\mathbf{a})$ as $\mathcal{N}(A, V)$. In that case, the decision variable can be transformed to:

$$y_k \propto \sqrt{e}a_k f(\mathbf{u},\mathbf{a}) + n_k \tag{4.52}$$

which gives rise to a bit error probability

$$E_{\mathbf{a},\mathbf{u}}\left[Q\left(\sqrt{f^2(\mathbf{u},\mathbf{a})\frac{2E_s}{N_0}}\right)\right] = Q\left(\sqrt{\frac{2eA^2}{2eV+1}}\right). \tag{4.53}$$

Substituting (4.53) into (4.51) and equating the arguments yields:

$$\frac{e_0}{e} = A^2 - 2e_0 V. \tag{4.54}$$

Replacing (4.54) in the definition (4.44) leads to

$$\text{BERdeg} = -10 \log \left(A^2 - V 2e_0 \right) \tag{4.55}$$

which is valid for $2e_0 V \ll 1$ and for small degradations (less than 1 dB).

In most cases $\sigma\left(\mathbf{u}\right)$ takes on values closely clustered around its mean, so that we may use the following approximation:

$$A = E_{\mathbf{a},\mathbf{u}}\left[f\left(\mathbf{u},\mathbf{a}\right)\right] \approx \frac{E_{\mathbf{a},\mathbf{u}}\left[f_N\left(\mathbf{u},\mathbf{a}\right)\right]}{E_{\mathbf{u}}\left[\sigma\left(\mathbf{u}\right)\right]} \tag{4.56}$$

and

$$\begin{aligned} V &= Var_{\mathbf{a},\mathbf{u}}\left[f\left(\mathbf{u},\mathbf{a}\right)\right] && (4.57) \\ &\approx \frac{E_{\mathbf{a},\mathbf{u}}\left[f_N^2\left(\mathbf{u},\mathbf{a}\right)\right]}{E_{\mathbf{u}}\left[\sigma^2\left(\mathbf{u}\right)\right]} - A^2. && (4.58) \end{aligned}$$

Chapter 5

Factor graphs, equalization and data detection

5.1 Introduction

In this chapter we will illustrate the measures a receiver needs to take in order to cope with adverse channel effects. Within our framework of factor graphs, we discern two stages: in the first stage the incoming continuous-time signal ($r_{IF}(t)$ or $r_{BB}(t)$) is transformed to a suitable observation $\mathbf{y}$. In the second stage, a factor graph of an a posteriori distribution is created and the sum-product algorithm is applied to yield a posteriori probabilities of the information bits. The first stage involves techniques such as matched filtering, whitening, sampling, timing correction. In the second stage, the problem of equalization will appear. Tasks such as synchronization and equalization are common to all receivers and have therefore received intense interest from the research community for over 50 years. Traditionally, equalization amounted to sampling and filtering the incoming signal in such a way that the channel looks more like a frequency-flat channel. The resulting samples were then provided to a decoder. Pioneering work in this area dates to the late 60s and early 70s [23,47,48]. In the next two decades, the theoretical aspects related to equalization have not received much attention until the advent of turbo codes in 1993 [7]. Only then it was recognized that information need not only flow from the equalizer to the decoder. By performing iterative equalization and decoding, performance can be improved significantly [18]. A good overview of such turbo-equalization techniques is given in [24]. The same paper also touches on the relationship between iterative equalization and factor graphs.

This chapter is organized as follows: we first define the ultimate goal of the receiver (i.e., data detection) and show how this goal may be achieved in two stages. The first stage is described in section 5.3 and concerns the conversion of the incoming signal to a suitable observation model. Secondly, from this observation model a factor graph can be constructed. When we apply the sum-product algorithm on this graph, the data can be recovered. Several practical ways to proceed are provided in section 5.4

5.2 Receiver operation

5.2.1 The received signal revisited

We start again from our received baseband signal. Revisiting Eq. (2.12):

$$r_{BB}(t) \quad = \quad \sum_{k=0}^{N_s-1} a_k h(t-kT) + w(t) \tag{5.1}$$

where $\sqrt{E_s}$ is the transmit energy per symbol, $w(t)$ is a Gaussian noise process that is white within the signal bandwidth and $h(t)$ is obtained by convolving the transmit pulse $p(t)$ with $h_{BB}(t)$

$$h(t) = \sqrt{E_s}\int_{-\infty}^{+\infty} p(u)\, h_{BB}(t-u)\, du. \tag{5.2}$$

Finally, $h_{BB}(t)$ is modeled as follows:

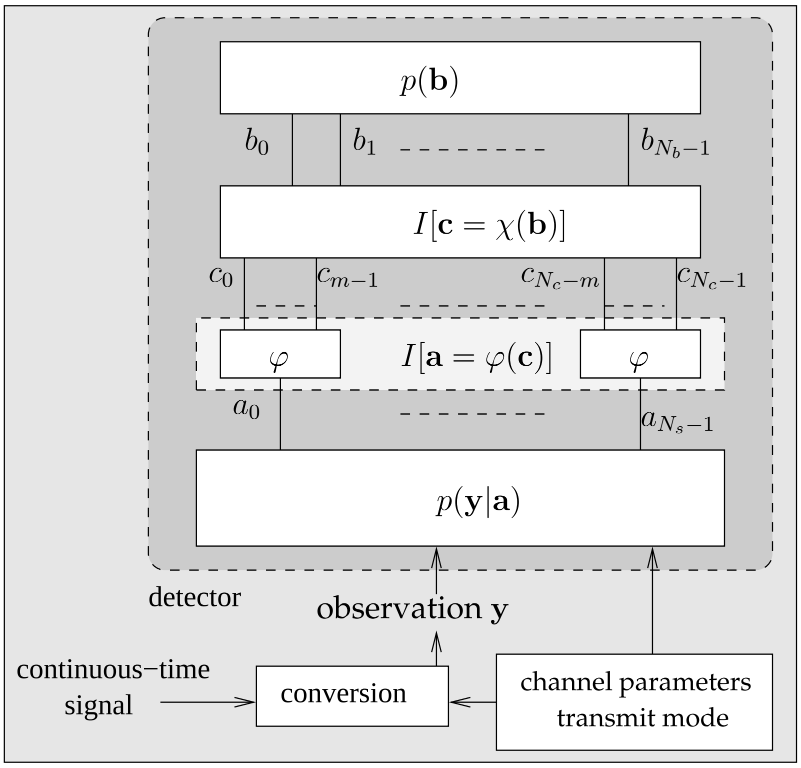


Figure 5.1: *Data detection: factor graph of* $p(\mathbf{b})\, I[\mathbf{c}=\chi(\mathbf{b})]\, I[\mathbf{a}=\varphi(\mathbf{c})]\, p(\mathbf{y}|\, \mathbf{a})$. *The observation* $\mathbf{y}$, *the channel parameters and transmit mode are* parameters *(not variables) in this graph.*

$$h_{BB}(t)=\sum_{l=0}^{L-1}\alpha_l\delta(t-\tau_l) \tag{5.3}$$

where α_l and τ_l are the complex gain and the propagation delay of the l-th path. When $L>1$, the channel is frequency-selective. When $L=1$, the channel is frequency-flat.

5.2.2 Data detection: principle

Our goal is to detect the sequence of information-bits $\mathbf{b}$ based on $r_{BB}(t)$. Here, $\mathbf{b}$ is related to $\mathbf{a}$ through $\mathbf{a}=\varphi(\mathbf{c})$, where $\mathbf{c}=\chi(\mathbf{b})$. χ represents the transformation from information-word $\mathbf{b}$ to codeword $\mathbf{c}$ and φ corresponds to the mapping of $\mathbf{c}$ to a modulation-set. Detection consists of two parts:

1. We first process the received signal $r_{BB}(t)$ to obtain an observation vector $\mathbf{y}$, so that the *likelihood function* $p(\mathbf{y}\,|\mathbf{a})$ has a form that can be cast in the suitable factor graph framework.

2. We construct the factor graph of the a posteriori distribution

$$p(\mathbf{b},\mathbf{a},\mathbf{c}|\,\mathbf{y})\propto p(\mathbf{b})\, I[\mathbf{c}=\chi(\mathbf{b})]\, I[\mathbf{a}=\varphi(\mathbf{c})]\, p(\mathbf{y}|\,\mathbf{a})\,. \tag{5.4}$$

   as depicted in Fig. 5.1. We then apply the sum-product algorithm on this factor graph, yielding the marginal a posteriori probabilities (APPs) $p(b_k|\,\mathbf{y})$. Decisions w.r.t. the information bits can be taken as follows

$$\hat{b}_k=\arg\max_{b_k} p(b_k|\,\mathbf{y})\,. \tag{5.5}$$

The remainder of this chapter is devoted to see how the transformation from $r_{BB}(t)$ into a suitable observation $\mathbf{y}$ is accomplished, what the corresponding factor graphs look like, and how the sum-product algorithm can be applied. The key to many detection algorithms lies in the factorization of the likelihood function $p(\mathbf{y}|\,\mathbf{a})$, with the introduction of additional variables. (see also 'the key idea' of MAP detection on factor graphs from page 30).

5.3 Observation models

We say that an observation $\mathbf{y}$ forms a *sufficient statistic* to detect the data sequence $\mathbf{b}$ when [46]

$$p(\mathbf{b}\,|\mathbf{y}) = p(\mathbf{b}\,|\mathbf{r}_{BB}) \tag{5.6}$$

where $\mathbf{r}_{BB}$ is obtained by projecting $r_{BB}(t)$ onto a suitable basis.

We will consider three observation models. A factor graph of the corresponding likelihood functions $p(\mathbf{y}\,|\mathbf{a})$ is constructed in section 5.4 for the three observation models. These likelihood functions can simply be 'plugged into' the factor graph from Fig. 5.1, so that the APPs can be computed. Although all three models happen to form a sufficient statistic to detect $\mathbf{b}$, we could also consider a myriad of sub-optimal observation models, each with a corresponding likelihood function (e.g., [49]). The principle for data detection remains the same: a factor graph will be constructed from the likelihood function and the SP algorithm is then used to detect the data sequence $\mathbf{b}$.

5.3.1 Observation model 1: matched filter detector

We denote by $g(t) = \int_{-\infty}^{+\infty} h^*(-u)\,h(t-u)\,du$. Expanding $r_{BB}(t)$ onto some orthonormal basis yields a vector representation $\mathbf{r}_{BB}$. As the noise at the input of the receiver is white within the signal bandwidth, we have the following likelihood-function:

$$\begin{aligned}
&\log p(\mathbf{r}_{BB}|\,\mathbf{a}) \\
&\propto -\int_{-\infty}^{+\infty}\left|r_{BB}(t) - \sum_{k=0}^{N_s-1} a_k h(t-kT)\right|^2 dt && (5.7)\\
&\propto 2\Re\left\{\sum_{k=0}^{N_s-1} a_k \int_{-\infty}^{+\infty} r_{BB}^*(t)\,h(t-kT)\,dt\right\} - \mathbf{a}^H\mathbf{G}\mathbf{a} && (5.8)\\
&= 2\Re\left\{\mathbf{y}_{MF}^H\mathbf{a}\right\} - \mathbf{a}^H\mathbf{G}\mathbf{a} && (5.9)
\end{aligned}$$

where $\mathbf{G}$ is a $N_s \times N_s$ Toeplitz matrix with

$$\mathbf{G}_{k,k'} = g((k-k')T) \tag{5.10}$$

and $\mathbf{y}_{MF} = [y_{MF,0},\ldots,y_{MF,N_s-1}]^T$ denotes the vector of N_s samples, obtained by filtering the baseband signal $r_{BB}(t)$ with a filter, $h^*(-t)$:

$$y_{MF,k} = \int_{-\infty}^{+\infty} r_{BB}(t)\,h^*(t-kT)\,dt. \tag{5.11}$$

The filter $h^*(-t)$ is referred to as the *matched filter* (i.e., matched to $h(t)$), while the samples $\{y_{MF,k}\}$ are known as the matched filter outputs. It is easily verified that $p(\mathbf{a}|\,\mathbf{r}_{BB}) = p(\mathbf{a}|\,\mathbf{y}_{MF})$, so that data detection can be performed, based solely on the N_s matched filter outputs $\mathbf{y}_{MF}$. The matched filter outputs can be written as

$$\mathbf{y}_{MF} = \mathbf{G}\mathbf{a} + \mathbf{n} \tag{5.12}$$

with

$$E\left[\mathbf{n}\mathbf{n}^H\right] = N_0\mathbf{G}. \tag{5.13}$$

Example

When the channel is frequency-flat, $h_{BB}(t) = Ae^{j\theta}\delta(t-\tau)$, where A is an amplitude, θ the carrier phase and τ the propagation delay. In that case, $h(t) = \sqrt{E_s}Ae^{j\theta}p(t-\tau)$, so that

$$\begin{aligned}
g(t) &= \int_{-\infty}^{+\infty} h^*(-u)\,h(t-u)\,du && (5.14)\\
&= A^2E_s\int_{-\infty}^{+\infty} p^*(-u-\tau)\,p(t-u-\tau)\,du && (5.15)\\
&= A^2E_s q(t) && (5.16)
\end{aligned}$$

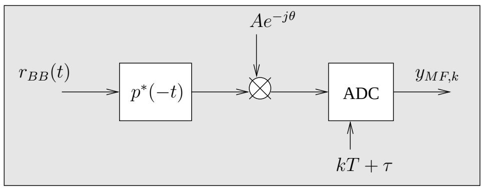


Figure 5.2: *Matched filter receiver for a frequency-flat channel*

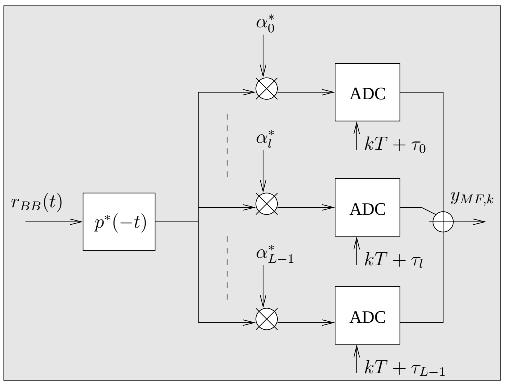


Figure 5.3: *Matched filter receiver for a Multi-path channel*

where

$$q(t) = \int_{-\infty}^{+\infty} p^*(-u)\, p(t-u)\, du. \tag{5.17}$$

When $p(t)$ is a square root Nyquist pulse,

$$g(nT) = A^2 E_s \delta_n \tag{5.18}$$

so that $\mathbf{y}_{MF} = A^2 E_s \mathbf{a} + \mathbf{n}$, with $E\left[\mathbf{n}\mathbf{n}^H\right] = A^2 E_s N_0 \mathbf{I}_{N_s}$. A receiver that computes the matched filter samples is shown in Fig. 5.2. This receiver is easily extended to a general multi-path scenario with $L > 1$ paths, as depicted in Fig. 5.3.

5.3.2 Observation model 2: whitening matched filter

When $g(t)$ is not proportional to a Nyquist pulse, detection based directly on the matched filter outputs may be difficult (since $\mathbf{G}$ in (5.10) will not be diagonal, so that the noise samples at the output of the matched filter are correlated). Fortunately, based on a spectral decomposition of the matrix $\mathbf{G}$, the noise at the output of the matched filter can be whitened (the details of whitening are omitted here, but can be found in any standard textbook on Digital Communications, such as [21]). By clever choice of the whitening filter, the resulting observation model corresponds to a causal channel with L_w taps[1]:

$$\mathbf{y}_{WMF} = \mathbf{H}\mathbf{a} + \mathbf{w} \tag{5.19}$$

with $E\left[\mathbf{w}\mathbf{w}^H\right] = N_0 \mathbf{I}_{N_s+L_w-1}$ and $\mathbf{H}$ an $(N_s + L_w - 1) \times N_s$ Toeplitz matrix. For instance, for a three-tap channel ($L_w = 3$) and $N_s = 4$, we get:

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & 0 & 0 \\ h_1 & h_0 & 0 & 0 \\ h_2 & h_1 & h_0 & 0 \\ 0 & h_2 & h_1 & h_0 \\ 0 & 0 & h_2 & h_1 \\ 0 & 0 & 0 & h_2 \end{bmatrix}. \tag{5.20}$$

[1]We note that there are many ways to whiten the noise in $\mathbf{y}_{MF}$ (e.g., through a Cholesky decomposition of $\mathbf{G}$, or a Karhunen-Loeve expansion). However, not all techniques will lead to a sufficient statistic. For more details on whitening, see [46].

5.3.3 Observation model 3: over-sampling detector

In the last technique, we proceed as follows: the incoming baseband signal $r_{BB}(t)$ is filtered with an analog anti-aliasing filter (AAF) and sampled at a rate N/T satisfying the Baseband Sampling Theorem. By suitable selection of the AAF, the noise of the samples will be white, iid and distributed according to $\mathcal{CN}(0, N_0 N/T)$. The samples can be written as

$$\mathbf{y}_{OS} = \mathbf{Ha} + \mathbf{w} \tag{5.21}$$

with $E\left[\mathbf{ww}^H\right] = N_0 N/T\mathbf{I}$. In the matrix $\mathbf{H}$, the row corresponding to sample $y_{OS}(mT + lT/N)$ is equal to (for $m \geq 0$):

$$\begin{array}{ccccccc} & & & \text{column } m & & & \\ & & & \downarrow & & & \\ [& \ldots & h\left(\frac{l+N}{N}T\right) & h\left(\frac{l}{N}T\right) & h\left(\frac{l-N}{N}T\right) & \ldots &]. \end{array}$$

Hence, the row corresponding to $y_{OS}((m+1)T + lT/N)$ is equal to the row corresponding to $y_{OS}(mT + lT/N)$, right-shifted over N columns. This regular structure is due to the fact that the sample rate is an integer multiple of the symbol rate. In principle, the incoming signal could be sampled at a rate that is not an exact multiple of the symbol rate. In such cases, possibly all rows in $\mathbf{H}$ will be different.

When the sample rate is an integer multiple of the symbol rate, it is readily verified that (5.21) can also be written as

$$\mathbf{y}_{OS} = \mathbf{Ah} + \mathbf{w} \tag{5.22}$$

where $\mathbf{h}$ contains the channel impulse response (assuming $h(t)$ takes on significant values in $[-L_l T, +L_r T]$):

$$\mathbf{h}^T = \left[h(L_r T)\ h(L_r T - T/N)\ h(L_r T - 2T/N) \ldots h(-L_l T + T/N)\ h(-L_l T)\right]$$

and the row of $\mathbf{A}$ corresponding to $y(mT + lT/N)$ is equal to

$$\begin{array}{cccccccccc} & & & \text{column } k_0 & & & & \text{column } k_0 + N & & \\ & & & \downarrow & & & & \downarrow & & \\ [0 & \ldots & 0 & a_0 & 0 & \ldots & 0 & a_1 & 0 & \ldots] \end{array}$$

where $k_0 = (L_r - m)N - l \geq 0$:

Remarks

- $\mathbf{y}_{OS}$ in (5.21) is a vector of length (roughly) $(N_s + L_h)N$, where L_h is the number of symbol periods for which $h(t)$ takes on significant values[2]. When the transmit pulse $p(t)$ takes on significant values for L_p symbol durations, then $L_h \approx L_p + \frac{\tau_{L-1} - \tau_0}{T}$.
- Depending on the channel, either a matched filter or an oversampling approach will be used. It is obvious that a matched filter receiver is to be preferred when the overall channel impulse response $h(t)$ closely resembles a square root Nyquist pulse. An oversampling receiver would in such cases entail a significant computational overhead. On the other hand, when $h(t)$ has a more exotic shape, an oversampling approach may be more suitable: it removes the need of explicit matched filtering and whitening.

5.4 Equalizers

Based on the three observation models (5.12), (5.19) and (5.21), we will now construct factor graphs of the corresponding likelihood functions $p(\mathbf{y}|\mathbf{a})$. The sum-product algorithm can then be applied on the factor graph from Fig. 5.1.

The sum-product algorithm may be too complex to implement in practice. For this reason we also describe a type of *augmented* equalizer: this is a block that replaces the node $p(\mathbf{y}|\mathbf{a})$ in Fig. 5.1, and accepts messages $\mu_{a_k \to \varphi}(a_k)$ and outputs messages $\mu_{\varphi \to a_k}(a_k)$, but *does not operate according to the sum-product algorithm.* As long as the augmented equalizer satisfies the input-output relationship of a node in a factor graph, it can be used instead of the sum-product algorithm in Fig. 5.1. Numerical results comparing the different equalizers can be found in abundance in technical literature [18,24,25,50,51] and are omitted here.

[2]Note that, strictly speaking, $h(t)$ is of infinite duration.

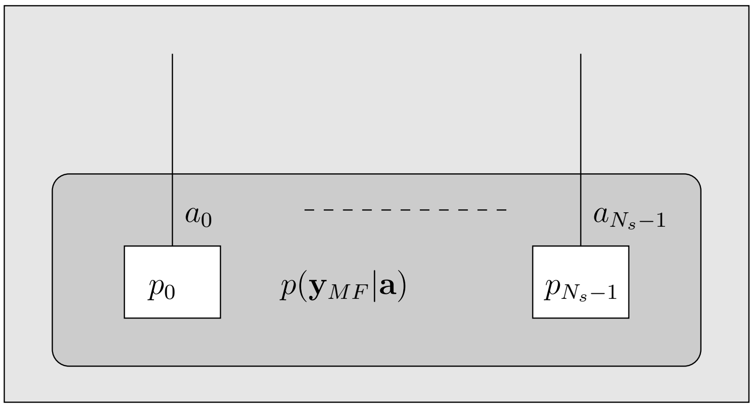


Figure 5.4: *Factor graph of likelihood function for Observation Model 1: frequency-flat channel*

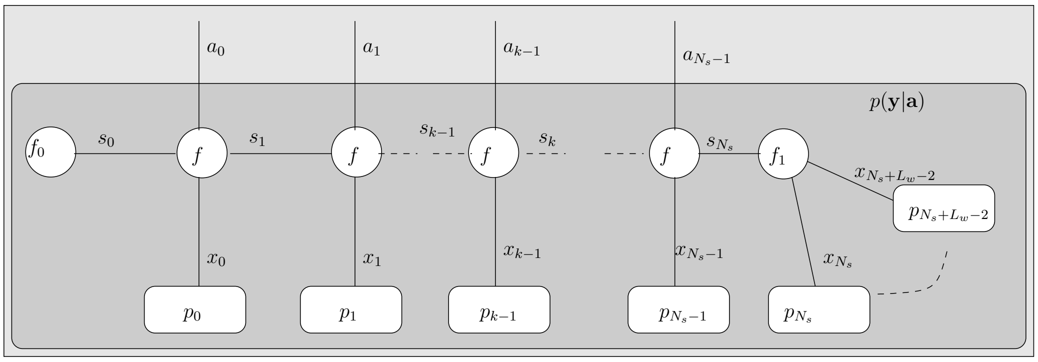


Figure 5.5: *Factor graph of likelihood function for Observation Model 2: frequency selective channel*

5.4.1 Sum-product equalizers

Observation model 1: matched filter detector

Observation model (5.12) generally gives rise to a likelihood function $p\left(\mathbf{y}_{MF}\middle|\,\mathbf{a}\right)$ that is not suitable in a factor graph representation, due to the correlation of the noise. Only in the case when $g\left(t\right)$ in (5.10) is a Nyquist pulse (e.g., when the channel is frequency-flat) can we obtain a suitable likelihood function of the following form

$$p\left(\mathbf{y}_{MF}\middle|\,\mathbf{a}\right) = \prod_{k=0}^{N_s-1} p\left(y_{MF,k}\middle|\,a_k\right) \tag{5.23}$$

$$\propto \prod_{k=0}^{N_s-1} \exp\left(-\frac{1}{A^2 E_s N_0}\left|y_{MF,k} - a_k A^2 E_s\right|^2\right). \tag{5.24}$$

The corresponding factor graph is shown in Fig. 5.4, where $p\left(y_{MF,k}\middle|\,a_k\right)$ is abbreviated by p_k. The sum-product algorithm is trivial: messages $\mu_{a_k\to\varphi}\left(a_k\right) = p\left(y_{MF,k}\middle|\,a_k\right)$ are propagated upwards to the mapper nodes.

Observation model 2: whitening matched filter

In the case of a symbol-rate detector with $\mathbf{H}$ corresponding to an L_w-tap channel, we may write the k-th component of (5.19) as[3]

$$y_k = \sum_{l=0}^{L_w-1} h_l a_{k-l} + w_k \tag{5.25}$$

$$\doteq x_k + w_k \tag{5.26}$$

[3]For notational convenience, we omit the subscript WMF in this paragraph.

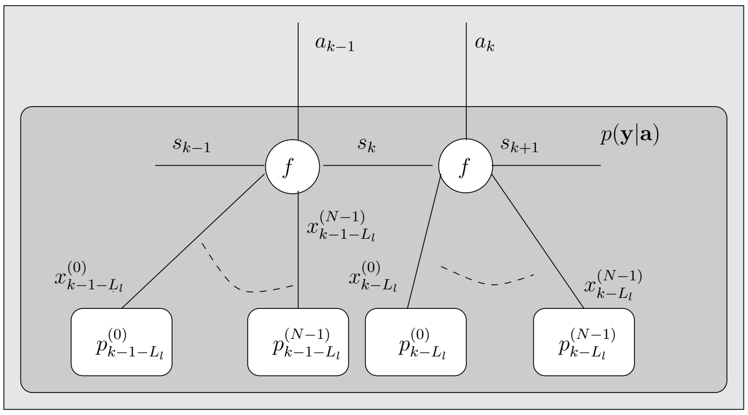


Figure 5.6: *Section of factor graph of likelihood function for Observation Model 3*

for $k = 0, \ldots, N_s + L_w - 2$. Introducing the additional variables $\{x_k\}_{k=0,\ldots,N_s+L_w-2}$ and $\{\mathbf{s}_k\}_{k=0,\ldots,N_s}$, with $\mathbf{s}_k = [a_{k-L_w+1}, \ldots, a_{k-1}]$, a vector of size $1 \times (L_w - 1)$ we can transform the likelihood function $p(\mathbf{y}|\,\mathbf{a})$ into the following factorization

$$p(\mathbf{y}, \mathbf{x}, \mathbf{s}|\,\mathbf{a}) = \prod_{k=0}^{N_s+L_w-2} p(y_k|\,x_k)\, I\left[x_k = h_0 a_k + f_{\mathbf{h}}(\mathbf{s}_k)\right] \tag{5.27}$$

where $f_{\mathbf{h}}(\mathbf{s}_k) = \sum_{l>0} h_l a_{k-l}$, and implicitly we assume $a_{<0} = 0$ and $a_{\geq N_s} = 0$. Abbreviating $p(y_k|\,x_k)$ with p_k, $I[x_k = h_0 a_k + f_{\mathbf{h}}(\mathbf{s}_k)]$ with $f(x_k, a_k, \mathbf{s}_k)$, and introducing

$$f_1(x_{N_s} \ldots, x_{N_s+L_w-2}, \mathbf{s}_{N_s}) \doteq \prod_{k=0}^{L_w-2} I\left[x_{k+N_s} = \sum_{l=1}^{L_w-1-k} h_{k+l} a_{N_s-l}\right]$$

and $f_0(\mathbf{s}_0) \doteq I\left[\mathbf{s}_0 = \mathbf{0}_{1\times(L_w-1)}\right]$, we have

$$p(\mathbf{y}, \mathbf{x}, \mathbf{s}|\,\mathbf{a}) = f_0(\mathbf{s}_0) \prod_{k=0}^{N_s-1} p_k f(x_k, a_k, \mathbf{s}_k) \prod_{k=N_s}^{N_s+L_w-2} p_k f_1(x_{N_s} \ldots, x_{N_s+L_w-2}, \mathbf{s}_{N_s}) \tag{5.28}$$

which lends itself well to a trellis representation, shown in Fig. 5.5.

Discussion

- Note that now the SP algorithm is applied to a factor graph representation of $p(\mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{x}, \mathbf{s}|\,\mathbf{y})$, rather than $p(\mathbf{b}, \mathbf{a}, \mathbf{c}|\,\mathbf{y})$.
- Observe the similarities between this factor graph and the one corresponding to the convolutional code generator block (Fig. 3.13). In the equalizer, the variables a_k are not binary (except for BPSK transmission), so the messages over the a_k-edges will be vectors of size $|\Omega|$. The variables x_k can take on $|\Omega|^{L_w}$ values. Hence, the complexity of the sum-product algorithm on this factor graph scales as $\mathcal{O}\left(N_s\,|\Omega|^{L_w}\right)$. For channels with large L_w, this approach is no longer suitable.
- When the channel is frequency-flat ($L_w = 1$), the state variables $\mathbf{s}_k$ are all of length $L_w - 1 = 0$, so that the factor graph from Fig. 5.5 reduces to the factor graph from Fig. 5.4.

Observation model 3: over-sampling detector

Let us focus on a situation where the incoming signal is sampled at T/N and the channel $h(t)$ takes on significant values in $(-L_l T, +L_r T)$, so that $L_h = L_l + L_r$. The equalizer will operate on the samples[4]

$$\mathbf{y} = \left[y(-L_l T), y(-L_l T + T/2), \ldots, y(N_s T - T + L_r T)\right]^T.$$

[4]For notational convenience, we omit the subscript OS in this paragraph.

Introducing the following notations $h_l^{(n)} = h(lT + nT/N)$, $n_l^{(n)} = n(lT + nT/N)$ for $n = 0, \ldots, N-1$, we can express $y_l^{(n)} = y(lT + nT/N)$ as

$$y_k^{(n)} = \sum_{l=-L_l}^{L_r} h_l^{(n)} a_{k-l} + n_k^{(n)} \tag{5.29}$$

where implicitely $a_{<0} = 0$ and $a_{\geq N_s} = 0$. After some straightforward manipulations:

$$\begin{aligned} y_{k-L_l}^{(n)} &= \sum_{l=0}^{L_r+L_l} h_{-L_l+l}^{(n)} a_{k-l} + n_{k+L_l}^{(n)} && (5.30) \\ &\doteq x_{k-L_l}^{(n)} + n_{k-L_l}^{(n)} && (5.31) \end{aligned}$$

which bears a striking resemblance to (5.25). The function $p(\mathbf{y}, \mathbf{x}, \mathbf{s}|\, \mathbf{a})$ can again be factorized in a similar fashion, and is represented by the factor graph depicted in Fig. 5.6. We have introduced the state variables $\mathbf{s}_k$, given by $\mathbf{s}_k = [a_{k-L_h}, \ldots, a_{k-2}, a_{k-1}]$. The nodes $p_k^{(n)}$ are a shorthand for $p\left(y_k^{(n)}\middle| x_k^{(n)}\right)$. The f-nodes ensure that $x_{k-L_l}^{(n)} = \sum_{l=0}^{L_r+L_l} h_{-L_l+l}^{(n)} a_{k-l}$, for $n = 0, \ldots, N-1$. The SP algorithm operates in the standard way. The complexity of this algorithm scales as $\mathcal{O}\left(N_s |\Omega|^{L_h}\right)$. Note that

- Each f-node in the factor graph from Fig. 5.6 has $3 + N$ connected edges (as opposed to at most 4 edges in the previous observation models);
- Some measures need to be taken at the boundaries of the factor graph (i.e., sections corresponding to $k = 0$ and $k = N_s - 1$), just like in the previous observation model;
- When the channel is frequency-flat, iterating between equalization and decoding is still necessary. In observation model 1 and observation model 2, iterating between equalization and decoding was not required as there were no cycles in the factor graph between the block $p(\mathbf{y}|\, \mathbf{a})$ and the block $I[\mathbf{a} = \varphi(\mathbf{c})]$.
- When the sample rate is not a integer multiple of the symbol rate, the function f may vary from time k to time $k+1$, which is clearly undesirable.

From the exposition above, it has become clear that a straightforward factor graph approach to equalization is possible, but computationally demanding when the channel is long (i.e., large L_w or L_h).

5.4.2 Augmented equalizers

Although the factor graph equalizers are interesting from a theoretical point of view, they are generally too complex to implement. Various ways to reduce the complexity have been proposed in technical literature. For instance, we could reduce the size of the state-space in the graphs from Figs. 5.5-5.6 by making decisions (either hard or soft) on the state. This reduces the complexity to $\mathcal{O}(N_s |\Omega|)$, irrespective of the number of taps. Such an approach was considered in [52].

Another way to proceed, is to look into the class of what we will call 'augmented' equalizers, where we replace the node corresponding to the likelihood function $p(\mathbf{y}|\, \mathbf{a})$ with a block that performs an algorithm, different from the sum-product algorithm. These augmented equalizers can be used for the three observation models and are basically conventional equalizers that have been augmented in three ways, so that they can operate within a factor graph:

1. **Input augmentation**: the equalizer can accept information from other parts of the factor graph, in particular messages regarding the symbols a_k: $\mu_{\varphi \to a_k}(a_k)$ (i.e., the message from the mapper node).
2. **Message computation**: messages should be computed in a way that does not violate the SP rule: an outgoing message $\mu_{a_k \to \varphi}(a_k)$ should not depend on the corresponding incoming message $\mu_{\varphi \to a_k}(a_k)$.
3. **Output augmentation**: the equalizer can output information in the form of messages $\mu_{a_k \to \varphi}(a_k)$, which are used elsewhere in the factor graph.

Various types of such augmented equalizers have been devised in technical literature. For the sake of illustration, we will describe a popular MMSE (Minimum Mean Squared Error) equalizer [49].

MMSE equalizer

The equalizer operates as follows. Suppose we want to estimate a quantity $\mathbf{a}$, with distribution $\mathcal{CN}\left(\mathbf{m}_a, \mathbf{S}_{aa}\right)$ from an observation $\mathbf{y}$, with

$$\mathbf{y} = \mathbf{H}\mathbf{a} + \mathbf{n} \tag{5.32}$$

where $\mathbf{n} \sim \mathcal{CN}\left(0, \mathbf{S}_{nn}\right)$. If we model the coded data symbols as Gaussian random variables, we can apply standard techniques (see [16, p. 643]) to estimate these symbols, for any of the three observation models (i.e., for (5.19), (5.21) as well as (5.12)):

$$\begin{aligned} \hat{\mathbf{a}} &= \mathbf{m}_a + \mathbf{S}_{ay}\mathbf{S}_{yy}^{-1}\left(\mathbf{y} - \mathbf{m}_y\right) & (5.33) \\ &= \mathbf{m}_a + \mathbf{S}_{aa}\mathbf{H}^H\left(\mathbf{H}\mathbf{S}_{aa}\mathbf{H}^H + \mathbf{S}_{nn}\right)^{-1}\left(\mathbf{y} - \mathbf{H}\mathbf{m}_a\right) & (5.34) \end{aligned}$$

where $\mathbf{S}_{ar}$ is the cross-covariance of $\mathbf{a}$ and $\mathbf{y}$, $\mathbf{S}_{yy}$ is the auto-covariance of $\mathbf{y}$ and $\mathbf{m}_y$ is the mean of $\mathbf{y}$. The components of $\hat{\mathbf{a}}$ are then estimates of the different symbols a_k. When information from the decoder is disregarded, $\mathbf{m}_a = \mathbf{0}$ and $\mathbf{S}_{aa} = \mathbf{I}_{N_s}$, in which case (5.34) reduces to a conventional MMSE equalizer [21]:

$$\hat{\mathbf{a}} = \mathbf{H}^H\left(\mathbf{H}\mathbf{H}^H + \mathbf{S}_{nn}\right)^{-1}\mathbf{y}. \tag{5.35}$$

Augmentation

In order to meet the three requirements above, the following modifications need to be made:

1. **Input augmentation**: the input messages $\mu_{\varphi \to a_k}\left(a_k\right)$ are used to approximate $\mathbf{m}_a$ and $\mathbf{S}_{aa}$: for instance, the k-th component of $\mathbf{m}_a$ would be given by[5] $m_{a,k} = \sum_{\omega \in \Omega} \omega \times \mu_{\varphi \to a_k}\left(\omega\right)$, while $E\left[a_k a_l^*\right]$ would be approximated by $m_{a,k}m_{a,l}^*$.
2. **Message computation**: when estimating a_k, we need to remove the dependence of the estimate $\hat{a}_k$ on the incoming message $\mu_{\varphi \to a_k}\left(a_k\right)$. This is achieved by replacing the corresponding entries in $\mathbf{m}_a$ and $\mathbf{S}_{aa}$ with those of the conventional MMSE equalizer [49].
3. **Output augmentation**: the MMSE equalizer outputs an estimate of each of the symbols. To convert these estimates to messages, we model the estimates $\hat{a}_k$ to have a Gaussian distribution with mean $\mu_k a_k$ and variance σ_k^2. Both μ_k and σ_k^2 can easily be determined from the available data [49].

Computational complexity

In practice, the computational complexity can be reduced by making use of a sliding-window approach: when determining $\hat{a}_k$, the MMSE equalizer operates only on $\mathbf{y}_k$, a subset of $\mathbf{y}$. Applying suitable approximations, the computational complexity is given by

- Observation Models 1 and 2: $\mathcal{O}\left(N_s L_w\right)$, for an L_w-tap channel
- Observation Model 3: $\mathcal{O}\left(N_s N L_h\right)$, for an $L_h N$-tap channel, with N samples per symbol duration.

When we compare with the factor-graph approach, we see a drastic reduction in complexity: the MMSE equalizer is no longer dependent on the constellation and its complexity is linear (rather than exponential) in the channel length.

5.5 Main points

In this chapter, we have discussed how a (mono-mode) receiver that has perfect knowledge of the channel state and synchronization parameters, can detect the transmitted data. First the received signal $r_{BB}\left(t\right)$ is transformed into a suitable observation $\mathbf{y}$. This involves tasks such as matched filtering, whitening of noise, sampling and timing correction. Secondly, by the introduction of additional variables (say, $\mathbf{x}$), a factor graph of the function $p\left(\mathbf{y}, \mathbf{x} \left| \mathbf{a}\right.\right)$ is created, and connected to the factor graph that models the a priori distribution of the coded data symbols $\mathbf{a}$. Applying

[5]Here, we exploit the fact that messsages are represented by pmfs.

the sum-product algorithm in this graph yields the APPs of the information bits. It turns out that, depending on the observation model and channel impulse response, we may need to iterate between decoding and equalization (turbo-equalization) using the SP algorithm. Unfortunately, this SP approach is not feasible for long channels. A second structure is mentioned, based on a conventional MMSE equalizer that is augmented to operate within the overall factor graph and replaces the node corresponding to $p\left(\mathbf{y},\mathbf{x}\left|\mathbf{a}\right.\right)$.

When the channel state changes, this may impact the conversion from $r_{BB}\left(t\right)$ to $\mathbf{y}$ as well the factor graph of $p\left(\mathbf{y},\mathbf{x}\left|\mathbf{a}\right.\right)$. In the latter case, this simply boils down to replacing a node in the factor graph. In the former case, depending on the observation model and type of channel, new filters may need to be loaded to perform the conversion from $r_{BB}\left(t\right)$ to $\mathbf{y}$. The complexity related to this heavily depends on the transmission model (e.g., CDMA, OFDM), and should be considered on a case-by-case basis. For the simple models we have considered, a matched filter receiver is better suited for frequency-flat channels, while an oversampling receiver may be preferred for frequency-selective channels.

Finally, we remind that any APPs computed in the detection process are actually conditioned on the (estimate of) channel state and transmit mode.

Chapter 6

Transmit mode adaptation

6.1 Introduction

Now that we have familiarized ourselves with some techniques related to matched filtering, equalization and data detection, we are ready to go one step further and consider the problem of transmit mode adaptation. In conventional mono-mode communications the transmitter does not have much freedom in the transmission. Basically, it can only change the information bit-sequence from frame to frame. In multi-mode transmission, the transmitter can set a number of parameters for each burst. The problem of mode adaptation is related to designing flexible transceivers in an efficient manner. At the same time we should stay true to the Software Radio paradigm and not allow any modes to depend on analog components. The latter goal is generally easy to achieve, since most modes are inherently digital. An exception is the shape of the transmit pulse, which lingers somewhere between the analog and digital domain. This is related to the critical aspect of symbol rate adaptation. We pay special attention to this problem, and place it in its common context, namely that of Direct-Sequence/Spread Spectrum (DS/SS) communications.

In parallel to the previous chapter, we assume that both the channel state and transmit mode can change from burst to burst, but remain constant within a burst. We again assume the receiver has perfect knowledge of the channel state and the transmit mode parameters. Our goal in this chapter is to see how a change in the transmit mode affects the receiver.

This short chapter is organized as follows: we first discuss the general problem of transmit mode adaptation in section 6.2 and then focus on the specific issues related to symbol rate adaptation in section 6.3. Symbol rate adaptation turns out to be far from trivial and will be covered in detail in the next chapter.

6.2 Transmit Mode Parameters

6.2.1 System Model

We repeat the transmission model from Chapter 2. The transmitter sends a burst of N_b information bits $\mathbf{b}$ to the receiver. The bits are first encoded, yielding a code sequence $\mathbf{c}$ of length N_c, $\mathbf{c} = \chi\left(\mathbf{b}\right)$. The ratio N_b/N_c is referred to as the code rate. The coded bits are then mapped to a sequence of N_s complex symbols $\mathbf{a} = \varphi\left(\mathbf{c}\right)$, with symbols taken from an M-point constellation Ω. The symbols are shaped by a unit-energy transmit pulse $p\left(t\right)$. The resulting complex baseband signal can be written as

$$s_{BB}\left(t\right) = \sqrt{E_s}\sum_{k=0}^{N_s-1} a_k p\left(t-kT\right) \tag{6.1}$$

where E_s and $1/T$ denote the transmit energy per symbol and the symbol rate, respectively. The transmit pulse has a (one-sided) bandwidth B.

In conventional (mono-mode) communications, the transmitter has only one degree of freedom in the creation of $s_{BB}\left(t\right)$: the sequence of information bits $\mathbf{b}$. Once $\mathbf{b}$ is set, the signal $s_{BB}\left(t\right)$ is fixed. In multi-mode transmission, the transmitter has the possibility to set additional parameters, according to a predefined set of rules. These parameters may include the transmit energy, the code, the constellation and the symbol rate. Each of these parameters affects the receiver in a different way.

The receiver operates on the signal $r_{BB}\left(t\right)$, obtained by filtering $s_{BB}\left(t\right)$ with the channel impulse response, and adding AWGN. In Fig. 6.1, we show the general structures we have derived so-far: the incoming signal $r_{BB}\left(t\right)$ is first processed to yield a suitable observation $\mathbf{y}$. A factor graph of $p\left(\mathbf{a},\mathbf{b},\mathbf{c}\left|\mathbf{y}\right.\right)$ is constructed. The parameters of

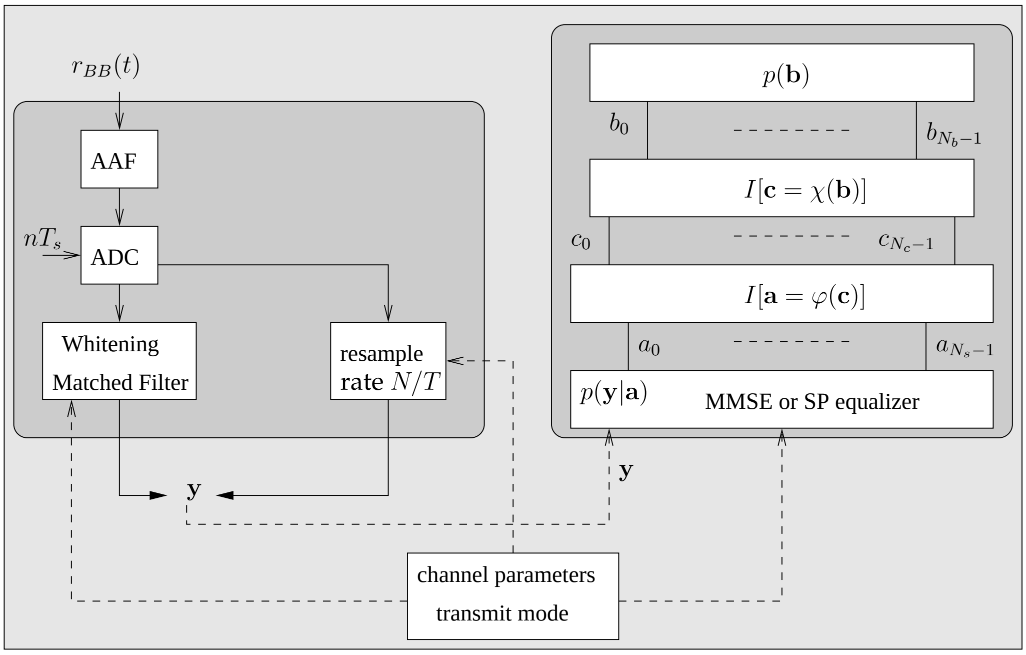


Figure 6.1: *Receiver: transformation from $r_{BB}(t)$ to observation $\mathbf{y}$. Factor graph of $p(\mathbf{a}, \mathbf{b}, \mathbf{c} | \mathbf{y})$.*

this graph are the observation $\mathbf{y}$, the channel parameters and the transmit mode. Applying the SP algorithm on this factor graph yields the a posteriori probabilities (APPs) of all variables. The node corresponding to $p(\mathbf{y} | \mathbf{a})$ may be replaced by a suitable block (e.g., an augmented MMSE equalizer).

6.2.2 Modes

Contrary to the previous chapter, transmit mode adaptation concerns both the transmitter and the receiver. We will now give a short overview of how modes may be changed and how they affect the transmitter and the receiver. As we will see, mode adaptation generally boils down to modifying nodes in the factor graph, i.e., a pure software change.

Transmit energy

The transmitter can change the transmit energy E_s in a limited continuous range. At the receiver, adapting to a change in E_s is easy: it corresponds to a trivial change in the node corresponding to the likelihood function $p(\mathbf{y} | \mathbf{a})$.

Code and code rate

By changing the error-correcting code χ and/or the code rate N_b/N_c, the transmitter is able to offer certain packets improved protection against adverse channel effects. At the receiver, the correct decoder must be plugged into the factor graph to decode the packet correctly. For codes with simple building blocks (such as convolutional codes), the code can be modified by changing the building blocks. For other codes (such as LDPC codes), this is not so straightforward. An efficient way to change the code rate is through *puncturing*. The process of puncturing was already briefly mentioned in Chapter 3: we fix the underlying code, but transmit only a (variable) subset of the coded bits over the channel. A related topic is that of rate-compatible punctured codes [53, 54], where code rates

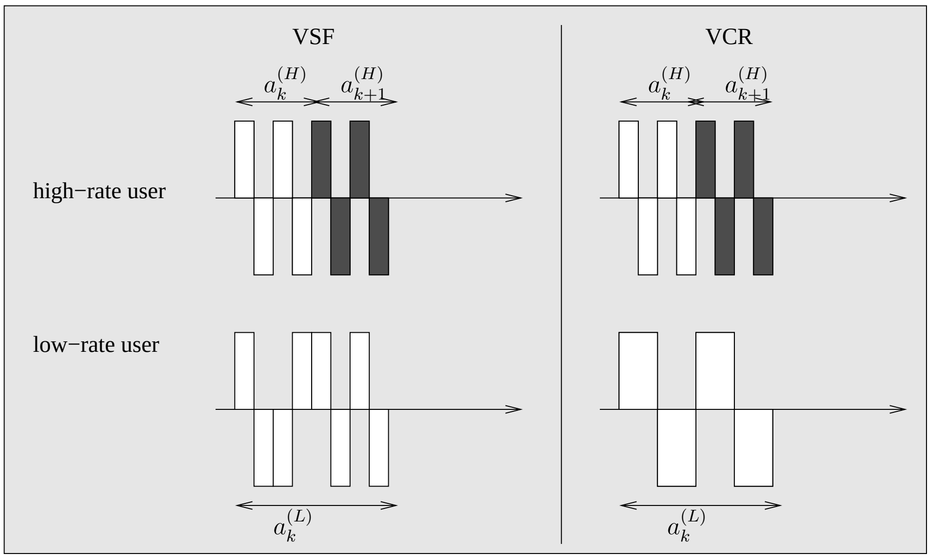


Figure 6.2: *Multi-rate DS/SS with Variable Spreading Factors (left) and Variable Chip Rates (right). For VCR the chip pulse of the low rate user (L) is longer than for the high-rate user (H).*

are organized in a hierarchy, so that higher-rate codes are embedded in low-rate codes. Finally, adaptive coded modulation has been investigated [55].

Constellation and mapping

Changing the constellation requires changes to the mapper block and (for factor-graph equalizers) to the equalizer block[1]. Changing the mapping (i.e., how a group of bits is mapped to a constellation point) requires only changes to the mapper block $I[\mathbf{a} = \varphi(\mathbf{c})]$. The latter is accomplished by storing a list of all possible mappings. Additional information on adaptive modulation can be found in [10, 56–59].

Symbol rate and pulse shape

This is probably the most complex task for the receiver: when the symbol rate changes, the pulse shape $p(t)$ changes accordingly. At the transmitter, this requires modifications to the digital filters used to generate $s_{BB}(t)$ from (6.1). At the receiver, changes are more drastic and pertain to the matched filter, the equalizer and the ADC. This is closely related to the concept of multi-rate transmission and will be discussed in more detail in the next section.

6.3 Symbol rate adaptation

The problem of symbol rate adaptation and multi-rate transmission is well known in the context of Direct-Sequence Spread Spectrum (DS/SS) systems. In such systems, the transmit pulse is given by[2]

$$p_T(t) = \sum_{n=0}^{N_g-1} \alpha_n p_{c,T_c}(t - nT_c) \tag{6.2}$$

where α_n is a chip, N_g is the spreading factor and $p_{c,T_c}(t)$ is a rate $1/T_c$ square root Nyquist chip pulse with one-sided bandwidth B_T. Note that $p_T(T)$ has the same bandwidth. The symbol rate is related to T_c and N_g by

$$1/T = 1/(N_g T_c). \tag{6.3}$$

Hence, the symbol rate can be increased by either reducing the number of chips per symbol or increasing the chip rate (or a combination of both). This leads to techniques known as Variable Spreading Factor (VSF) and Variable Chip Rate (VCR) multi-rate transmission [60–62] (see Fig. 6.2).

[1]Note that an MMSE equalizer requires no knowledge of the constellation.
[2]The subscript T will indicate the symbol duration.

Let us compare these two techniques on a simple example: multi-rate transmission over a frequency-flat channel $h_{BB}(t) = Ae^{j\theta}\delta(t-\tau)$. The corresponding matched filter is given by (see section 5.3)

$$h^*(-u) = Ae^{-j\theta}\sum_{n=0}^{N_g-1}\alpha_n^* p_{T_c}^{*,chip}(-u-nT_c-\tau) \tag{6.4}$$

which can be implemented in the following 4 steps[3]:

1. Filtering with a chip-matched filter $p_{c,T_c}^*(-t)$, yielding a signal $x_1(t)$ where

$$\begin{aligned} x_1(t) &= Ae^{j\theta}\sqrt{E_s}\sum_{k=0}^{N_s-1} a_k \sum_{n=0}^{N_g-1}\alpha_n \int p_{c,T_c}^*(-u)\, p_{c,T_c}(t-u-nT_c-\tau-kT)\,du \\ &\doteq Ae^{j\theta}\sqrt{E_s}\sum_{k=0}^{N_s-1} a_k \sum_{n=0}^{N_g-1}\alpha_n q_{c,T_c}(t-nT_c-\tau-kT) \end{aligned}$$

2. Timing correction to compensate for τ, yielding samples (assuming an adjustable clock)

$$\begin{aligned} x_1(lT+mT_c+\tau) &= Ae^{j\theta}\sqrt{E_s}\sum_{k=0}^{N_s-1} a_k \sum_{n=0}^{N_g-1}\alpha_n q_{c,T_c}(lT-kT+mT_c-nT_c) \\ &= Ae^{j\theta}\sqrt{E_s}\sum_{k=0}^{N_s-1} a_k \sum_{n=0}^{N_g-1}\alpha_n \delta_{l-k}\delta_{m-n} \\ &= Ae^{j\theta}\sqrt{E_s}a_l\alpha_m \end{aligned}$$

3. Despreading, yielding samples $x_2(lT)$, with

$$\begin{aligned} x_2(lT) &= \sum_{m=0}^{N_g-1}\alpha_m^* x_1(lT+mT_c+\tau) \\ &= Ae^{j\theta}\sqrt{E_s}a_l\sum_{m=0}^{N_g-1}|\alpha_m|^2 \end{aligned}$$

4. Carrier phase correction

VSF multi-rate transmission

In a VSF system the rate can be reduced by increasing the spreading factor. The maximal symbol rate equals the chip rate (corresponding to $N_g = 1$), with each symbol corresponding to a single chip. From the description of the matched filter, we see that as the spreading factor changes, this only has an impact on step 3. With step 3 being a simple correlation that is implemented digitally, a change in the spreading factor can easily be adapted to.

In the context of multi-user systems, the difficulties related to VSF are the following: first of all, VSF requires careful dynamic allocation of spreading codes to each of the users. Secondly, there is no way to change the bandwidth corresponding to different rates. Also, only very few users can be supported at the highest data rate.

VCR multi-rate transmission

In VCR transmission, the symbol rate is reduced by using a low-rate chip pulse, but maintaining the same spreading factor. Looking again at the 4 steps in matched filtering, a change in the chip pulse impacts step 1, requiring a new filter to be loaded for each rate.

In the context of multi-user systems, VCR has the benefit of superior performance compared to VSF [63] and has more freedom in terms of the number of users that can be supported at a given data rate. However, VCR transmission has some practical drawbacks: since low-rate users do no use up the same bandwidth as high-rate users, VCR requires advanced frequency planning. Secondly, since DS/SS systems commonly use RAKE receivers (similar to the receiver depicted in Fig. 5.3) without any further equalization, low-rate users will not be able to

[3]We will neglect the noise in this paragraph.

	transmitter	receiver
E_s	trivial	trivial
code/code rate	code FG	code FG
modulation set	mapper FG + code FG	mapper FG + code FG + equalizer FG
modulation mapping	mapper FG	mapper FG
T, pulse	digital filters	digital + analog filters + equalizer FG

Table 6.1: *Modifications required at receiver and transmitter to support multi-mode transmission. FG stands for factor graph.*

exploit multi-path diversity to the same extent as high-rate users. This problem could be countered by allowing low-rate users to perform additional equalization. Finally, a third concern with VCR is that the matched filter needs to be changed in a non-trivial way according to the symbol rate: for each rate, a new chip-pulse $p_{c,T_c}(t)$ would need to be loaded at both the transmitter and the receiver. This is hardly convenient. The next chapter is devoted to how multi-rate transmission with varying bandwidths can be implemented efficiently by removing the need to store multiple impulse responses of the rate-dependent chip pulses. Note that VCR can also be applied to systems without spreading.

6.4 Main points

We have given a brief overview of which parameters can be changed during multi-mode transmission (see table 6.1). Similarly to the previous chapter, two types of adaptation can be discerned: adaptation in the transformation of the incoming signal $r_{BB}(t)$ to a suitable observation $\mathbf{y}$ and adaptation of the factor graph. It turns out that most modes can be easily adapted to, by changing the corresponding part in the receiver's factor graph. Only the symbol rate and pulse shape cause some practical problems, since they are related to the transformation of the incoming signal. In particular, a matched filter receiver needs to load new filter taps each time the symbol rate changes. An oversampling receiver must change the sampling rate if it wishes to operate at a rate that is a fixed multiple of the symbol rate. This calls for sample rate conversion techniques. As we will see later, the matched filter receiver also suffers from a dependency of the computational complexity on the symbol rate. In the next chapter we will tackle these problems.

Chapter 7

Low-Complexity receivers

7.1 Introduction

In the previous chapter, we have shown how a receiver can adapt, at least in principle, to the transmit mode. However, there are a number of practical concerns that were not discussed thoroughly, related to the conversion of the incoming continuous-time signal ($r_{BB}(t)$ or $r_{IF}(t)$) to a suitable observation $\mathbf{y}$. More specifically, in the context of Software Radio it is desirable to perform tasks as much as possible in the digital domain [14, 64]. Any analog processing that is mode-dependent should be removed and replaced with a digital counterpart. For instance, rather than sampling the output of the matched filter, we could sample its input at some suitable sampling rate and perform matched filtering digitally. Although shifting operations to the digital domain leads to a multi-mode receiver that is fully realizable in software, there are still a number of issues we need to contend with.

First of all, we would like filters to operate (as much as possible) at a rate that is independent of the symbol rate. A related issue is that of Sample Rate Conversion (SRC): we would like components to operate at an integer multiple of the symbol rate. Since we can not guarantee the sampling rate to be an exact integer multiple of all (or even any) symbol rates, SRC is required. This problem is present in both the matched filter receiver and the oversampling receiver [65]. Finally, we could like to go one step further and sample the IF signal directly. This has a number of benefits that will be detailed later on in this chapter. However, sampling the IF signal directly results in high-frequency components that may interfere with the useful signal. Care needs to be taken in the design of such an IF-sampling receiver.

These modifications will result in a receiver that closely adheres to the 'everything digital' Software Radio paradigm. The receiver will operate almost fully in the digital domain at low complexity. However, these modifications may have a significant impact on the performance of the receiver. These issues will be the topic of the current chapter. We will focus on the matched filter receiver, although many points are valid for the oversampling receiver.

This chapter is organized as follows: in section 7.2 we give a qualitative description of some receiver alternatives. The problem of sample rate conversion is described in section 7.3. The bulk of this chapter is found in section 7.4 and is devoted to the evaluation of different receiver alternatives in terms of computational complexity and BER performance. We end with some remarks regarding coded systems in section 7.5.

7.2 Design issues in multi-rate receivers

The matched filter receiver from Chapters 5-6 assumed an adjustable clock that samples the signal at the output of the matched filter at the correct time instants, driven by a synchronizer (analog timing correction). The corresponding receiver is shown in Fig. 7.1: analog IF-to-baseband conversion and analog matched filtering is followed by synchronized symbol-rate sampling. The analog matched filter in this receiver configuration can be replaced by an equivalent structure with a digital matched filter[1] (in which case the synchronized baseband sampling in front of the matched filter is at a multiple of the symbol rate, and the matched filter output is decimated to the symbol rate). In addition, also the IF-to-baseband conversion can be performed digitally by using synchronized bandpass sampling (in which case the intermediate frequency and the sampling rate must be carefully selected to avoid aliasing from the double-frequency terms [26]). The advantage of IF-sampling as compared to baseband sampling is that IF-to-baseband conversion is performed digitally, so that the need for analog quadrature oscillators, identical analog low-pass filters and identical analog to digital converters in the in-phase and quadrature branches is avoided [66].

[1]Provided the sampling rate satisfies the Baseband Sampling Theorem (see Chapter 4).

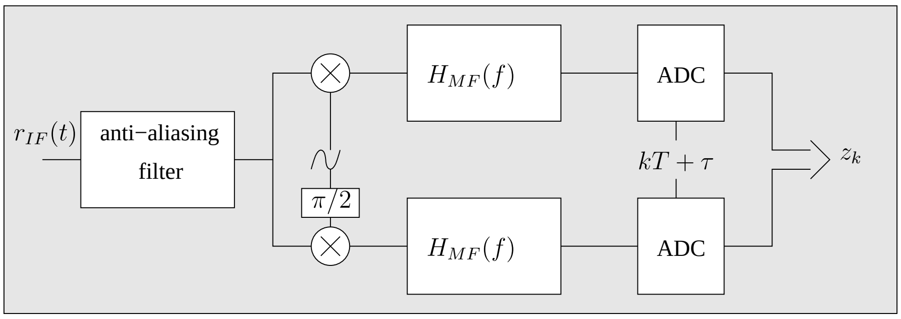


Figure 7.1: *Reference receiver structure with analog matched filtering (MF) and synchronized sampling*

Analog timing correction can be replaced with digital timing correction, where sampling is performed by means of a fixed free-running clock, and synchronized samples are obtained by interpolating between the available non-synchronized samples [39, 40]. The synchronized instants at which an interpolant is needed are determined by a synchronizer, and forwarded to the interpolator. From the implementation point of view, digital timing correction is to be preferred, because an inexpensive fixed sampling clock can be used. On the other hand, because of its finite memory the interpolator introduces distortion that degrades the performance as compared to the receiver with synchronized sampling (Fig. 7.1). Different receiver configurations with digital timing correction can be envisaged: non-synchronized sampling can be performed either at baseband or at IF, and interpolation can either precede or follow matched filtering. The performance degradation caused by non-ideal interpolation in the case of baseband sampling has been investigated in [44, 45, 67].

In the current chapter, these receiver alternatives will be investigated in some detail.

7.3 Sample rate conversion

For low-cost designs (such as for wireless devices), the clock rate cannot be guaranteed to be a multiple of the symbol rates. On the other hand, it is often desirable to let digital filters operate at a multiple of the symbol rate. Among other things, this allows filters to be implemented using a polyphase approach [65].

For multi-rate transmission, there is another dimension to this problem. Suppose we have a system where we are able to sample the signal at a multiple of the smallest symbol rate (the 'base' rate). Assume further that all symbol rates are an integer multiple of the base rate. Note that the sampling rate has to satisfy the baseband sampling theorem for all symbol rates. This means that for low data rates, the signal will be heavily oversampled. In such a case computational complexity will strongly depend on the symbol rate: for high data rates computational complexity may be significantly smaller than for lower data rates. Ideally we would like computational complexity to be independent of the data rate.

These problems are all related to the concept of sample rate conversion, i.e., how to convert samples at a given rate to samples at another rate. The problem of sample rate conversion was considered in [15, 43, 65, 68–72]. These contributions deal mostly with fractional sample rate conversion (i.e., transforming from rate N_1/T to rate N_2/T). Due to the particular nature of the receivers we will consider, we need to allow for more general sample rate conversion.

7.4 Low-complexity matched filter receivers

We have described some critical issues in the design of digital multi-rate receivers:

- the digital implementation of the matched filter;
- in the case of digital timing correction: the location of the interpolator;
- IF-sampling;
- sample rate conversion.

Now, we are ready to embark on a more quantitative analysis. Our final goal is to create a receiver that operates, as much as possible, like a Software Radio, while at the same time not significantly impacting the overall performance.

Let us start from the receiver from Fig. 7.1. This receiver will be designated the 'reference receiver', as all our new receivers will be compared to it. We will now introduce four receivers that operate in the digital domain, whereby the received signal is sampled at a rate $1/T_s$, a rate that may be incommensurate with some or all of the symbol rates. These four receivers will be compared to the reference receiver in terms of BER performance at the output of the matched filter for BPSK modulation. To keep the analysis tractable, a frequency-flat channel is assumed with an overall channel impulse response given by $\delta(t-\tau)e^{j\theta}$ where τ is the propagation delay and θ the carrier phase. Both θ and τ are assumed to be known to the receiver.

The BER performance results we will obtain, will translate into simple design criteria such that the BER degradation of the four receivers compared to the reference receiver will be negligible (i.e., less than, say, 0.1 dB) for all symbol rates. While BER performance results were obtained for a very specific scenario (i.e., a very specific channel model and a very specific type of modulation), the resulting design criteria in no way depend on that scenario and are valid for more general cases, even for coded transmission.

We will proceed as follows: we first give a description of the receiver front-end, followed by a listing of four possible digital receivers. One of these receivers combines IF-sampling with symbol-rate independent matched filtering. We will then detail the effects of aliasing on such receivers, paying special attention to the role of the digital interpolators and the use of digital anti-aliasing filters. This is followed by a detailed performance analysis, based on the BER degradation of each of the four receivers, as compared to the reference receiver. We then present design criteria for each of the receivers. The resulting design rules are subsequently verified by numerical evaluation of the BER degradation and by computer simulations.

7.4.1 System description

7.4.1.1 Receiver front-end

Let us first re-visit the system model. We consider multi-rate burst transmission, where the symbol rate ($1/T$) is constant during a burst, but can change from one burst to the next. After propagation through the channel $\delta(t-\tau)e^{j\theta}$, we write the received IF signal as

$$r_{IF}(t) = s_{IF}(t,\tau,\theta) + n(t) \tag{7.1}$$

where $n(t)$ is real additive white Gaussian noise (AWGN) with power spectral density equal to $N_0/2$ and $s_{IF}(t,\tau,\theta)$ is the IF signal:

$$\begin{aligned} s_{IF}(t,\tau,\theta) &= \Re\left\{\sqrt{2E_s}\sum_k a_k p_T(t-kT-\tau)e^{j2\pi f_{IF}t}e^{j\theta}\right\} & (7.2)\\ &\doteq \Re\left\{\sqrt{2}s_{LP,T}(t-\tau)e^{j2\pi f_{IF}t}e^{j\theta}\right\} & (7.3)\end{aligned}$$

where f_{IF} is the IF, $\{a_k\}$ are the uncorrelated data symbols with $E\left[|a_k|^2\right]=1$, E_s is the energy per symbol, $p_T(t)$ is a square-root cosine-roll-off unit energy transmit pulse with roll-off $\alpha \in [0,1]$ and a one-sided bandwidth $B = (1+\alpha)/(2T)$, θ is the carrier phase and τ is the propagation delay of the complex envelope. We assume θ to be a random variable that is uniformly distributed in $[0,2\pi[$. Note that both τ and θ are constant for a given burst, but can vary from burst to burst. The symbol interval T can take on values within an interval $[T_{min}, T_{max}]$. Consequently, the bandwidth B of the transmit pulse is in a corresponding interval $[B_{min}, B_{max}]$. As in the previous two chapters, the carrier phase θ, the propagation delay τ and the IF f_{IF} are assumed to be known at the receiver.

The corresponding complex baseband signal can be obtained by down-conversion followed by low-pass filtering

$$r_{BB}(t) = s_{LP,T}(t-\tau)e^{j\theta} + w(t). \tag{7.4}$$

7.4.1.2 Receiver alternatives

The reference receiver from Fig. 7.1 can be used to detect the data symbols. Unfortunately, this receiver is far from a Software Radio receiver: almost all components are analog. Among other things, this implies that the analog matched filter would have to be replaced as the symbol rate changes.

We will describe four receiver alternatives. The first two are baseband (BB) sampling receivers and are depicted in Fig. 7.2. Then we will describe two IF-sampling receivers, depicted in Fig. 7.3.

Baseband sampling receivers The first BB receiver is used in most current receivers: the baseband signal is sampled with a free-running clock and matched filtered prior to timing correction. It is easily verified that when the

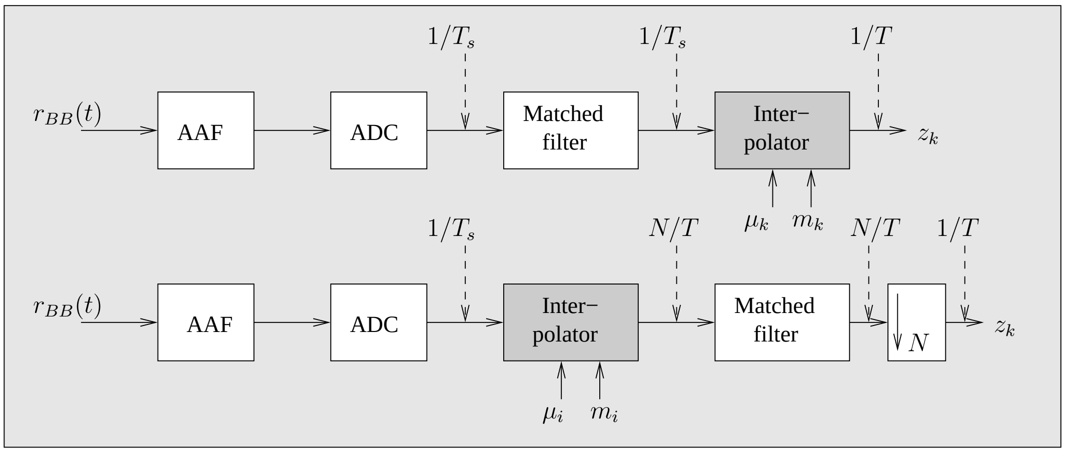


Figure 7.2: *Two baseband sampling seceivers. AAF stands for analog Anti-Aliasing Filter.*

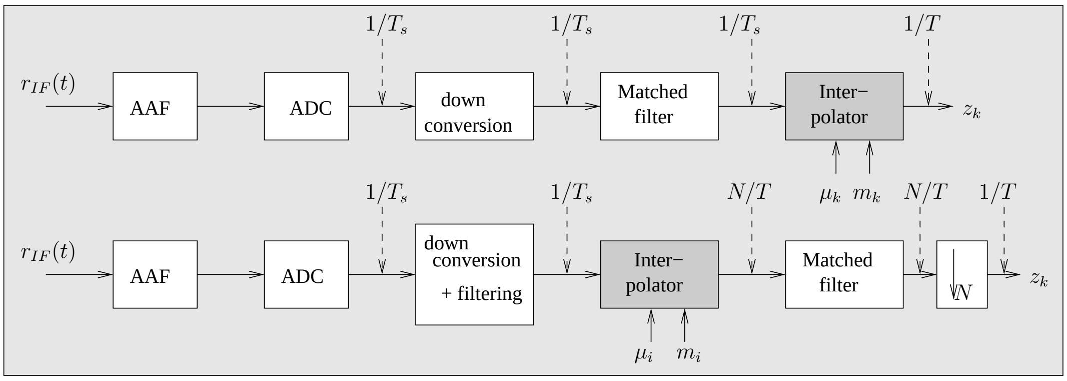


Figure 7.3: *Two IF-sampling receivers. AAF stands for analog Anti-Aliasing Filter.*

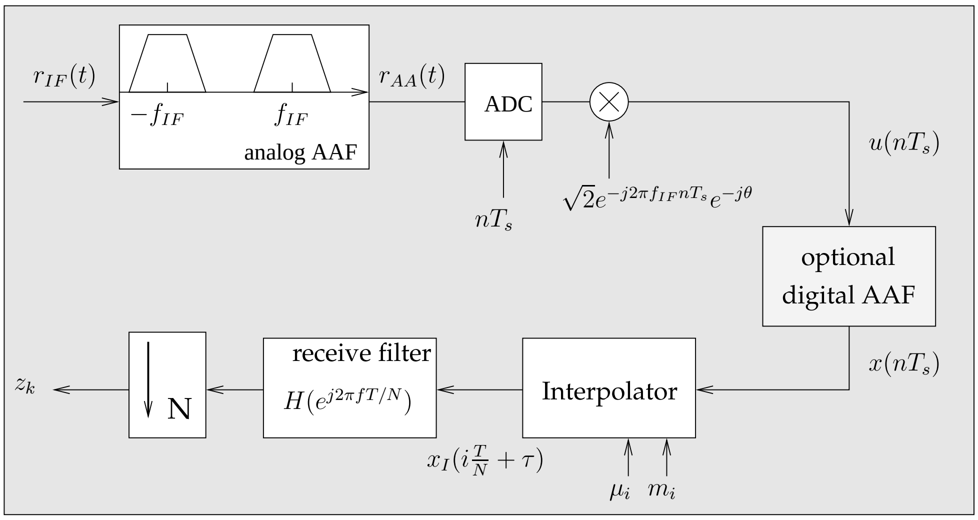


Figure 7.4: *Digital receiver with IF sampling and sample rate conversion before matched filtering*

sampling rate satisfies the baseband sampling theorem, and when the interpolator is ideal, this receiver has exactly the same performance as the reference receiver. Unfortunately, the taps of the matched filter will still need to be changed as the symbol rate varies: different sets of matched filter taps need to be stored and loaded depending on the symbol rate. Also, the computational complexity of the matched filter will be significantly larger for low data-rates.

The second receiver is obtained by interchanging the matched filter and interpolator: now the matched filter taps no longer depend on the symbol rate[2] (up to an irrelevant multiplicative constant). Only one set of taps needs to be stored at the receiver. Also, the computational complexity related to matched filtering is independent of the symbol rate. Hence, from the complexity point of view, the preferred configuration in a multi-rate receiver is to interpolate in front of the matched filter.

IF-sampling receivers These are very similar to the BB-sampling receivers, with the notable exception that down-conversion is now performed digitally, so that the need for analog quadrature oscillators, identical analog low-pass filters and identical analog-to-digital converters in the in-phase and quadrature branches is avoided. The receiver where interpolation precedes matched filtering is of particular interest: it combines all benefits of IF-sampling and low-complexity matched filtering. However, the high-frequency components that are now still present in the down-converted signal may interfere with the useful signal. In the sequel we will show that a receiver with interpolation in front of matched filtering is more susceptible to aliasing than is the receiver with matched filtering in front of interpolation. To overcome this, we include an optional digital anti-aliasing filter after down-conversion. As this receiver combines many aspects not present in conventional receivers, it will be our main focus during this chapter.

7.4.1.3 IF-sampling receiver: operation

We have proposed a receiver that combines IF-sampling with low-complexity multi-rate matched filtering. We will now describe this receiver in more detail. As shown in Fig. 7.4, the received signal $r_{IF}(t)$ is applied to an analog anti-aliasing bandpass filter, sampled by a free-running clock at a rate $1/T_s$, and frequency-translated from IF to baseband by applying a constant-speed rotation of $-2\pi f_{IF}T_s$ rad/sample. The anti-aliasing filter (AAF) is a fixed analog filter that does not change with the symbol rate of the received burst. We model the equivalent low-pass filter as a 4-th order analog Butterworth filter with 3 dB cut-off frequency equal to $B_{AA} = B_{max}(1+e)$, with $e > 0$. Note that the bandwidth of the noise at the output of the AAF is larger than the bandwidth of the useful signal, especially when operating at the minimum symbol rate. The AAF has a nearly flat frequency response for $|f|$ in the interval $[f_{IF} - B_{max}, f_{IF} + B_{max}]$, in order to avoid large distortions of the useful signal.

Subsequently, synchronized samples of the down-converted signal, taken at a multiple (N) of the symbol rate (i.e., at instants $iT/N + \tau$), are obtained by means of an interpolator (IP). The interpolator takes care of both timing correction and sample rate conversion. As interpolators have poor anti-aliasing properties, we also consider applying

[2]Under the assumption that for any T_1 and T_2: $p_{T_1}(nT_1/N) \propto p_{T_2}(nT_2/N)$. This assumption is valid for many types of transmit pulses.

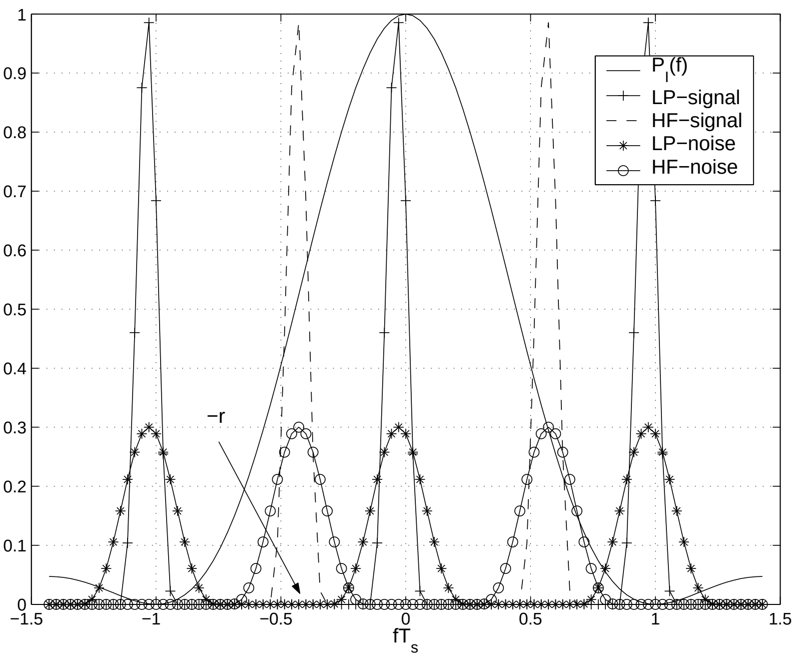


Figure 7.5: *Frequency-domain view of low-pass (LP) signal and noise (centered around $f = k/T_s$, for $k \in \mathbb{Z}$), high-frequency (HF) signal and noise (centered around $f = -r/T_s + k/T_s$, for $k \in \mathbb{Z}$) and $P_I(f)$, the Fourier transform of the linear interpolator.*

an optional digital AAF, placed before the interpolator [73]. This filter should be independent of the symbol rate and have a near-flat characteristic in the band $[-B_{max}, +B_{max}]$.

The resulting synchronized samples are applied to a discrete-time receive filter that is matched to the transmit pulse. In Fig. 7.4 this filter is represented through its DTFT $H\left(e^{j2\pi fT/N}\right)$. The matched filter (MF) output is decimated by a factor N, yielding samples z_k at the decision instants $kT + \tau$. The matched filter also suppresses (part of) the double-frequency terms (at $-2f_{IF}$ + multiples of $1/T_s$) that result from the down-conversion of the sampled output of the analog anti-aliasing filter. The matched filter output samples are fed to the decision device that detects the transmitted symbols $\{a_k\}$.

7.4.2 Aliasing

The bandpass signal $r_{AA}(t)$ at the output of the AA filter is sampled at rate $1/T_s$ and frequency translated (in discrete time) by an amount $-f_{IF}$. At the same time, carrier phase correction is performed. The resulting sequence $\{u(nT_s)\}$ can be interpreted as samples of a signal $u(t)$, which is related to $r_{AA}(t)$ and to the complex envelope $r_{AA,LP}(t)$ of $r_{AA}(t)$ by

$$\begin{aligned} u(t) &= \sqrt{2}r_{AA}(t)\, e^{-j2\pi f_{IF}t}e^{-j\theta} && (7.5)\\ &= r_{AA,LP}(t) + r^*_{AA,LP}(t)\, e^{-j4\pi f_{IF}t}e^{-j2\theta}. && (7.6)\end{aligned}$$

The Discrete-Time Fourier Transform (DTFT) of the sequence $\{u(nT_s)\}$ consists of a periodical extension (with period $1/T_s$) of $R_{AA,LP}(f) + R^*_{AA,LP}(-f - 2f_{IF})\, e^{-j2\theta}$, with $R_{AA,LP}(f)$ denoting the Fourier Transform (FT) of $r_{AA,LP}(t)$.

In order to avoid aliasing with the useful signal at the input of the interpolator, shifts (by a multiple of $1/T_s$) of the high-frequency (HF) component $R^*_{AA,LP}(-f - 2f_{IF})$ should not overlap with the low-pass component $R_{AA,LP}(f)$ within the interval $(-B, B)$. Defining[3] $r = rem(2f_{IF}T_s)$, the shifted HF components located closest to $f = 0$ are centered at $-r/T_s$ and $(1 - r)/T_s$. The periodical extension of the LP and HF components is shown in Fig. 7.5 for $r = 0.4$. Applying the Bandpass Sampling Theorem, we require that $(B_{AA} + B)\, T_s < \min(r, 1 - r)$ for all symbol rates. Hence, a sufficient condition to avoid aliasing at the input of the interpolator is as follows:

$$(B_{AA} + B_{max})\, T_s \le \min(r, 1 - r)\,. \qquad (7.7)$$

[3]$rem(x) \in [0, 1[$ denotes the fractional part of x.

Although condition (7.7) assures that no aliasing occurs at the *input* of the interpolator, aliasing at the *output* of the interpolator is still possible. An interpolator is able to remove the periodical extension of $R_{AA,LP}(f)$, but generally has poor anti-aliasing properties, i.e., the ability to remove the periodical extension of $R^*_{AA,LP}(-f-2f_0)$ [74] (this effect is clearly visible in Fig. 7.5). The latter may get aliased with the useful signal at the output of the interpolator and cause significant degradations.

By applying a simple digital anti-aliasing filter before the interpolator, the signal contributions that cause the aliasing may be (partially) removed. We will now describe the aspects of interpolation and anti-aliasing in more detail.

7.4.2.1 Interpolation

The interpolator output rate is an integer multiple (N) of the symbol rate $1/T$. Since the propagation delay τ is known, the values μ_i and m_i (as described in section 4.4, page 54), to be provided to the interpolator, are known at the receiver. In practice, these quantities need to be estimated by a timing recovery circuit. For the remainder of this chapter, we will use a two-tap linear interpolator in all examples and results. As higher order polynomial interpolators have similar Fourier transforms, generalization is straightforward. In fact, the mathematical derivations do not assume any particular type of interpolator.

7.4.2.2 Digital anti-aliasing filter

We denote the output of the digital AAF (DAAF) by $x(nT_s)$ (see Fig. 7.4). When no DAAF is present, $x(nT_s) \equiv u(nT_s)$. We consider two types of anti-aliasing filters: ideal low-pass filters and CIC filters.

Ideal anti-aliasing filter This DAAF is spectrally flat within $(-B_{max}, B_{max})$ and rejects frequency components outside $(-B_{max}, B_{max})$. When (7.7) is satisfied, there can be no aliasing from high frequency components within the signal bandwidth at the DAAF output. Although such a filter is not realizable in practice, its performance will serve as a reference.

CIC anti-aliasing filter As ideal low-pass filters are not realizable, we also consider a class of CIC filters. Decimating CIC filters are *low-complexity* decimation filters (with input rate $1/T_s$ and output rate $1/(RT_s)$) with good anti-aliasing properties. These filters were described in section 4.4. Unfortunately, in combination with an interpolator, the use of CIC filters has several side-effects: the interpolator has to work at a lower rate ($1/(RT_s)$ instead of $1/T_s$), reducing its performance. Secondly, CIC filters suffer from a severe passband 'droop', especially for high symbol rates. While the latter problem is tackled in [75], solutions to the former problem were discussed in [71] in the context of Software Radio receivers: using a polyphase filter structure allowed the interpolator to operate at the higher rate. This is similar to [72] where a set of R parallel CIC filters were placed before the interpolator. An extra control structure forwarded the correct samples to the interpolator. Here we propose a solution that is mathematically equivalent to [72], but more simple: by dropping the trailing decimator (see Fig. 4.6, page 56), the resulting filter, which we denote by $CIC(R,L)$, has equal input and output rates. It now serves solely as an anti-aliasing filter and not as a decimator. The frequency magnitude response of a $CIC(R,L)$ is given by (up to an irrelevant constant):

$$|H_{DAA}(f)| = \left|\frac{\sin(\pi f R T_s)}{\sin(\pi f T_s)}\right|^L \tag{7.8}$$

where the subscript DAA refers to the digital anti-aliasing filter. As can be seen in Fig. 7.6, a $CIC(R,L)$ has nulls for $fT_s = kR$, for $k \in \mathbb{Z}$, $0 \neq k \bmod R$ and is flat around $fT_s = k$. It can easily be shown that while increasing the number of stages in the CIC filter (i.e., the order, L) improves the alias rejection, it also increases the passband droop, thus distorting the useful signal. When we combine this filter with the frequency response of a polynomial interpolator, we end up with an interpolating filter that has very attractive anti-aliasing properties, especially if the HF signal components happen to be centered at nulls of the CIC filter, e.g., when $rT_s = kR$, $0 \neq k \bmod R$.

CIC implementation For a practical implementation, it is important to note that the overall impulse response can be re-written as follows [76]:

$$\begin{aligned} H(z) &= \left(\frac{1}{1-z^{-1}}\right)^L \times \left(1-z^{-R}\right)^L && (7.9)\\ &= \left(\sum_{k=0}^{R-1} z^{-k}\right)^L. && (7.10)\end{aligned}$$

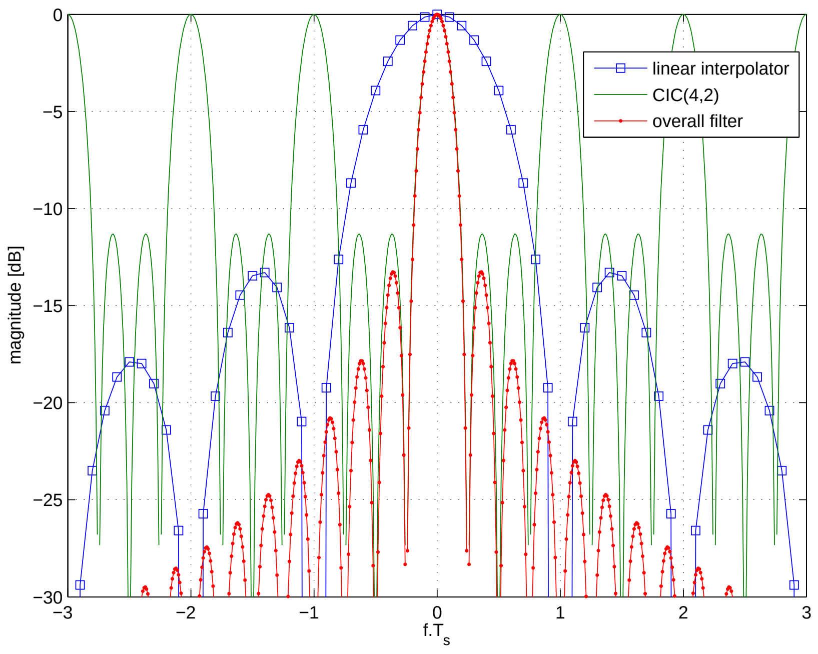


Figure 7.6: *Magnitude frequency response for $CIC\,(4,2)$ filter, a linear interpolator and the combined filter*

Hence, the CIC filter can be implemented by the L-fold concatenation of the FIR filter $1+z^{-1}+\ldots+z^{-R+1}$.

7.4.3 Performance measure

The performance measure we consider is the BER degradation compared to the reference receiver from Fig. 7.1. The decision variable is given by

$$z_k = \sqrt{E_s}a_k s_0\left(\mathbf{u}\right) + \sqrt{E_s}\sum_{n\neq k} a_n s_{k-n}\left(\mathbf{u}\right) + \sqrt{N_0/2}\sigma\left(\mathbf{u}\right)n_k \tag{7.11}$$

where $\mathbf{u} = [\tau, \theta]$. For an IF-sampling receiver, $s_k\left(\mathbf{u}\right)$ may contain contributions due to the high-frequency component of the signal (see Eq.(7.5)). Similarly, the noise can contain baseband and high-frequency components. We introduce the following notations:

- σ^2_{tot}: average of $\sigma^2\left(\mathbf{u}\right)$ (average w.r.t. noise statistics, carrier phase and decision instants),
- P_{LP}: average power of the baseband component in $s_0\left(\mathbf{u}\right) + \sum_{n\neq k} a_n s_{k-n}\left(\mathbf{u}\right)$ (average w.r.t. data symbols, carrier phase and decision instants),
- P_{HF}: average power of the high-frequency component in $s_0\left(\mathbf{u}\right)+\sum_{n\neq k} a_n s_{k-n}\left(\mathbf{u}\right)$ (average w.r.t. data symbols, carrier phase and decision instants),
- P_U: part of P_{LP} due to the current data symbol a_k.

Applying the BER degradation equation (4.55) from Chapter 4, with $V = \left(P_{LP}+P_{HF}\right)/\sigma^2_{tot} - A^2$ and $A^2 = P_U/\sigma^2_{tot}$, yields

$$\text{BERdeg} \quad = \quad -10\log\frac{1}{\sigma^2_{tot}}\left(P_U - \left(P_{LP}+P_{HF}-P_U\right)\left(\frac{2E_s}{N_0}\right)\right). \tag{7.12}$$

The reference receiver from Fig. 7.1 is not affected by HF terms, nor by inter-symbol-interference, so that $P_{LP} = P_U = 1$, $P_{HF} = 0$, $\sigma^2_{tot} = 1$, yielding a BER degradation of 0 dB. In the baseband sampling receivers, no high-frequency components are present in the signal or noise, so that $P_{HF} = 0$.

We will now evaluate z_k further for the four receivers. The evaluation is semi-analytical: averaging w.r.t. data symbols, noise statistics, decision instants and carrier phase is performed analytically by transforming z_k to a frequency-domain form. The resulting integrals in the frequency domain can then be evaluated numerically. This is in contrast with [44, 45, 67] where only averaging w.r.t. data symbols is performed analytically and all remaining averaging is performed through (time-consuming) computer computations.

We will introduce the following acronyms for the four receivers:

1. IP+MF:BB refers to baseband sampling, with interpolation before matched filtering,
2. IP+MF:IF refers to IF-sampling, with interpolation before matched filtering,
3. MF+IP:BB refers to baseband sampling, with matched filtering before interpolation ,
4. MF+IP:IF refers to IF-sampling, with matched filtering before interpolation.

Receiver (1) was proposed in [67]. Receiver (2) is the proposed low-complexity receiver from Fig. 7.4. Receiver (3) is the most common receiver in current digital communications systems, while receiver (4) is a common IF-sampling receiver.

7.4.4 Performance analysis

7.4.4.1 IP + MF

In order to investigate the effect of the non-ideal interpolation on the decision variable z_k, we will first relate the DTFT of the DAAF input samples to the DTFT of the MF input samples. Based on the latter, we can easily compute z_k.

Taking into account that the interpolator operates on samples $x(nT_s)$ of a signal $x(t)$, and applying (4.26)-(4.27), we obtain the following relationship between the DTFTs of $\{x_I(iT/N+\tau)\}$ and $\{u(nT_s)\}$:

$$X_I\left(e^{j2\pi f\frac{T}{N}}\right) = \frac{N}{TT_s}\sum_{m,n=-\infty}^{+\infty} Q_n\left(f-m\frac{N}{T}\right) \tag{7.13}$$

where

$$Q_n(f) = U\left(f-\frac{n}{T_s}\right)H_{DAA}\left(e^{j2\pi fT_s}\right)P_I(f)\,e^{j2\pi f\tau}. \tag{7.14}$$

$P_I(f)$ and $U(f)$ are the FT of the interpolating pulse $p_I(t)$ and of the signal $u(t)$ from (7.14), that underlies the samples $\{u(nT_s)\}$ at the input of the digital anti-aliasing filter. When no digital AA filter is present, we set $H_{DAA}\left(e^{j2\pi fT_s}\right)=1$. In (7.13) the summations over m and n reflect the sampling of $x_I(t)$ and $x(t)$ at rates N/T and $1/T_s$, respectively. We can now express z_k as

$$\begin{aligned} z_k &= \frac{T}{N}\int_0^{N/T} H\left(e^{j2\pi f\frac{T}{N}}\right)X_I\left(e^{j2\pi f\frac{T}{N}}\right)e^{j2\pi fkT}df && (7.15)\\ &= \frac{1}{T_s}\int_0^{N/T} H\left(e^{j2\pi f\frac{T}{N}}\right)\sum_{n=-\infty}^{+\infty}Q_n(f)\,e^{j2\pi fkT}df && (7.16)\end{aligned}$$

where $H\left(e^{j2\pi f\frac{T}{N}}\right)$ denotes the FT of the sequence $\{h[i]\}$ (i.e., the matched filter taps). This leads to

$$z_k = \int_0^{N/T} H\left(e^{j2\pi f\frac{T}{N}}\right)U\left(e^{j2\pi fT_s}\right)H_{DAA}\left(e^{j2\pi fT_s}\right)P_I(f)\,e^{j2\pi f(kT+\tau)}df \tag{7.17}$$

To see how, in the absence of a DAAF, the HF components in $U(f)$ contribute to (7.17) (and consequently increase the BER degradation), let us list the important factors in the integrand: $H\left(e^{j2\pi f\frac{T}{N}}\right)$, $P_I(f)$ and $U\left(e^{j2\pi fT_s}\right)$. Breaking up the latter factor in a low-pass (LP) and high-frequency (HF) component, we obtain a frequency-domain view, depicted in Fig. 7.7 (with $H_{DAA}\left(e^{j2\pi fT_s}\right)=1$, $r=0.6$ and $T/(NT_s)=2.22$). From the figure it is clear that HF

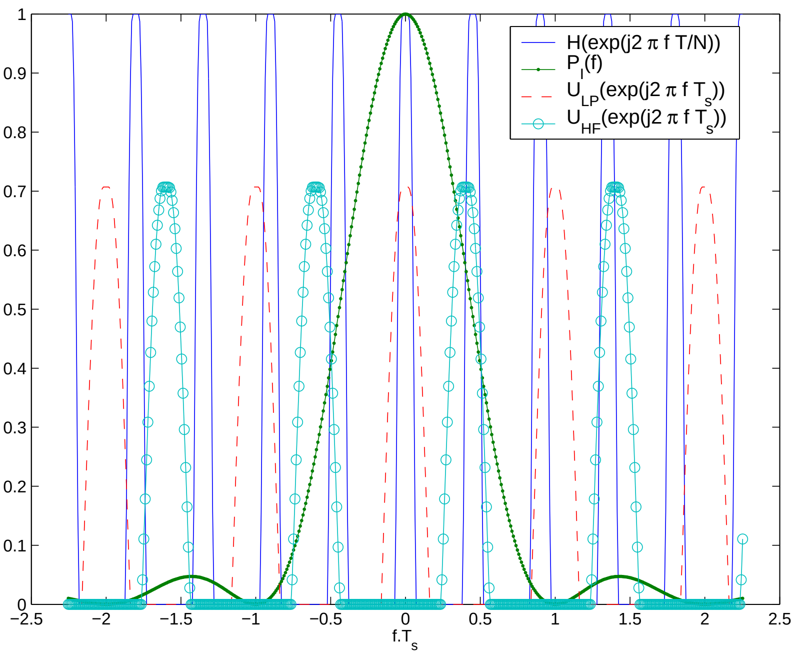


Figure 7.7: *Frequency-domain interpretation for Interpolator + Matched filter for* $r = 0.6$, $T/(NT_s) = 2.2200$.

components contribute to the integral (for instance the component at $fT_s = 0.4$ overlaps with the component of $H\left(e^{j2\pi f\frac{T}{N}}\right)$ centered at $fT_s = T_sN/T = 1/2.22 = 0.4505$). These components will cause a degradation if they are not removed by a digital AAF.

Hence, a HF component causes aliasing when it is close to the center of the periodic extension of the matched filter. This occurs when, for any $n, m \in \mathbb{Z}$:$-r/T_s + n/T_s$ is close to mN/T. Especially when mN/T is within the main lobe of $P_I(f)$ ($[-1/T_s, +1/T_s]$ in Fig. 7.7) aliasing can be significant.

7.4.4.2 MF + IP

In this case the matched filter rejects aliasing outside the useful signal bandwidth so that no separate digital AA filter is needed. Hence, z_k is expressed as

$$z_k = \int_0^{1/T_s} U\left(e^{j2\pi fT_s}\right) H\left(e^{j2\pi fT_s}\right) P_I(f)\, e^{j2\pi f(kT+\tau)} df. \tag{7.18}$$

Notice the subtle differences between (7.17) and (7.18). When (7.7) holds, the HF terms (with bandwidth B_{AA} and centered around $k/T_s - r/T_s$) contained in the sequence $\{u(mT_s)\}$ do not contribute to (7.18), as they are suppressed by the matched filter ($H\left(e^{j2\pi fT_s}\right)$), which has a bandwidth less than B_{max} and is centered around k/T_s, $k \in \mathbb{Z}$. The components of the integrand in (7.18) are depicted in Fig. 7.8. Now, provided (7.7) is satisfied, high-frequency signal components fall outside the matched filter bandwidth. Hence, they will not contribute to the integral.

7.4.5 Receiver Design Parameters

Now that we have determined the decision variables for each of the four receivers, we can compute the power of the low-pass and high-frequency signal and noise components. In principle, this is achieved by breaking up z_k into the four corresponding components and calculating their power. This calculation is straightforward but very tedious: it basically amounts to performing a great deal of integrations and summations and would cover many pages in this manuscript. As these exact computations do not yield any additional insights, they are omitted here. At this point, we have sufficient information to determine design criteria for our four receivers. Later, in section 7.4.6, we will evaluate these criteria in terms of their BER degradation.

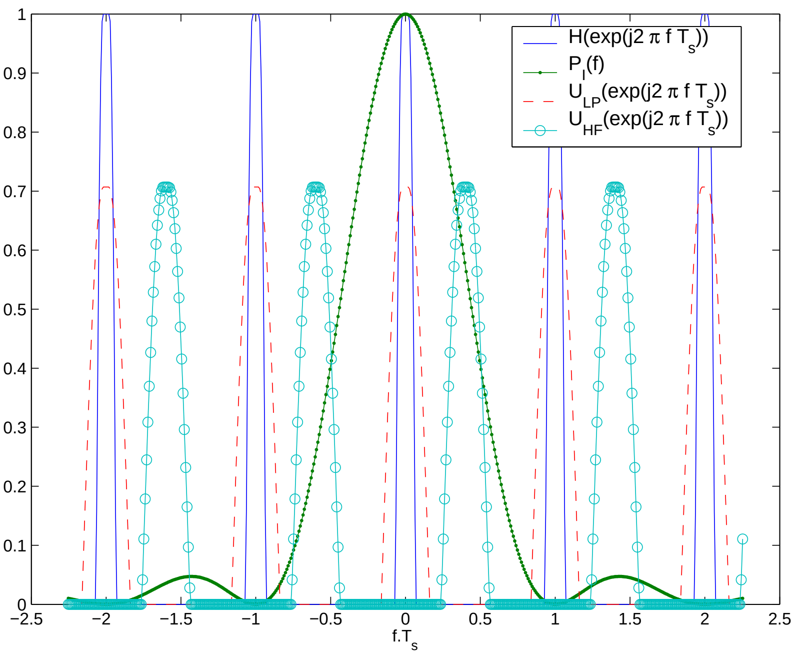


Figure 7.8: *Frequency-domain interpretation for Matched filter + Interpolator for $r = 0.6$. Observe that HF components fall outside the MF bandwidth.*

7.4.5.1 Baseband sampling receivers

MF+IP

We need to ensure no aliasing occurs within the useful signal bandwidth at the input of the matched filter. This is achieved when the Baseband Sampling Theorem is satisfied for all T:

$$\frac{1}{T_s} > B_{AA} + B_{max}. \tag{7.19}$$

A higher sample rate will improve interpolator performance and reduce the degradation with respect to the receiver from Fig. 7.1.

IP+MF

The same restriction (7.19) on T_s applies to avoid aliasing within the signal bandwidth at the interpolator *input*. To ensure that the signal in $[-B_{AA}, B_{AA}]$ at the interpolator input causes no aliasing within the signal bandwidth $[-B, B]$ at the interpolator *output*, we need (applying the Baseband Sampling Theorem again):

$$N > T_{max}\left(B_{min} + B_{AA}\right). \tag{7.20}$$

Under these restrictions degradations will be low. Again, a higher sampling rate $1/T_s$ will lead to better performance. N and T_s can be chosen independently. Note that for BB-sampling all considered values for N satisfying (7.20) lead to the same degradation.

7.4.5.2 IF-sampling receivers

MF+IP

In order to avoid that sampling at rate $1/T_s$ gives rise to aliasing within the signal bandwidth at the input of the matched filter, the sample rate and IF are restricted by (7.7):

$$\frac{1}{T_s} \geq \frac{B_{AA} + B_{max}}{\min\left(r, 1-r\right)}. \tag{7.21}$$

Increasing the sampling rate $1/T_s$ improves the interpolator performance and reduces the degradations with respect to the receiver from Fig. 7.1. When (7.7) holds, the performance is the same as for the MF+IP receiver with baseband sampling.

IP+MF without digital AAF

In [77] we have shown that the receiver performance degradation is very large when the dominating HF components (located at frequencies $(1-r)/T_s$ and $-r/T_s$) at the interpolator input get aliased into the useful signal bandwidth $[-B, B]$ at the interpolator output (see Fig. 7.7, where HF components at $fT_s = -0.6$ and $fT_s = 0.4$ undergo the least attenuation from $P_I(f)$). It was observed that low BER degradations (i.e., less than 0.1 dB for a BER of 10^{-3}) can be achieved by avoiding this aliasing by means of proper parameter selection. More specifically, these low degradations can be obtained when

$$\frac{1}{T_s} > \frac{B_{AA} + B_{max}}{1/2} \tag{7.22}$$

and

$$N \geq T_{max}\left(2B_{AA} + B_{max} + B_{min}\right). \tag{7.23}$$

Condition (7.22) assures that there exist values for r (implicitly given by (7.7)) so that there is no aliasing within the useful signal bandwidth at the input of the interpolator. Condition (7.23) assures that the dominating HF component is outside the useful signal bandwidth at the output of the interpolator. Defining $r^* = T_s\left(B_{max} + B_{AA}\right)$, it can be verified (see [77]) that aliasing due to the dominating HF component is then avoided only when $r \approx r^*$ or $r \approx 1 - r^*$. Furthermore, when

$$N \geq T_{max}\left(\frac{1}{2T_s} + B_{AA} + B_{min}\right) \tag{7.24}$$

both HF components are suppressed for $r \approx \frac{1}{2}$. Note that the oversampling factor N required in both (7.23) and (7.24) may be significantly larger than the one required in (7.20). This is clearly undesirable. Hence, such a receiver is of little practical interest for multi-rate systems.

IP+MF with ideal digital AAF

When the interpolator is preceded by a fixed digital AAF that suppresses frequencies outside the interval $[-B_{max}, +B_{max}]$, the parameters T_s, N and r must satisfy (7.20) and (7.7) in order to avoid aliasing from high-frequency components of $\{u(nT_s)\}$ within the signal bandwidth $[-B, B]$ at both the input and the output of the interpolator. When (7.7) holds, the resulting performance is equal to that of the IP+MF receiver with baseband sampling, irrespective of N.

IP+MF with CIC digital AAF

We can suppress signal components that are located near the nulls of a CIC filter (see section 7.4.2.2). By increasing either the order (L) or the factor (R) of the CIC filter, HF signal and noise components at the interpolator input are more attenuated. At some point, increasing L or R will not affect the influence of the HF components to the BER degradation (i.e., these components have become negligible). At the same time, due to the passband droop for $R > 1$, the useful signal is distorted, especially for high symbol rates (small T) and high CIC orders (large L). Hence, a CIC filter will reduce aliasing (which is good) but also distort the useful signal (which is bad). Clearly, both r and the CIC parameters (L and R) will play a key role in this trade-off.

We further remind that the receive filter with taps $\mathbf{h} = \{h[i]\}$(with DTFT $H(\exp(j2\pi fT/N))$) was assumed matched to the transmit pulse $p(t)$. Consequently, it is not matched to the entire received pulse (i.e., the concatenation of the transmit pulse, the CIC filter and the interpolator). It therefore makes sense to optimize the receive filter taps $\mathbf{h}$ in order to counteract the distortion of the useful component introduced by the interpolator and (especially) the CIC filter. We have selected the following optimization criterion: minimize (w.r.t. $\mathbf{h}$) the maximal (maximized over all considered T) Mean Squared Error (MSE) between a_k and z_k:

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}}\left\{\max_{T}\left\{E\left[\left|a_k - z_k\right|^2\right]\right\}\right\}. \tag{7.25}$$

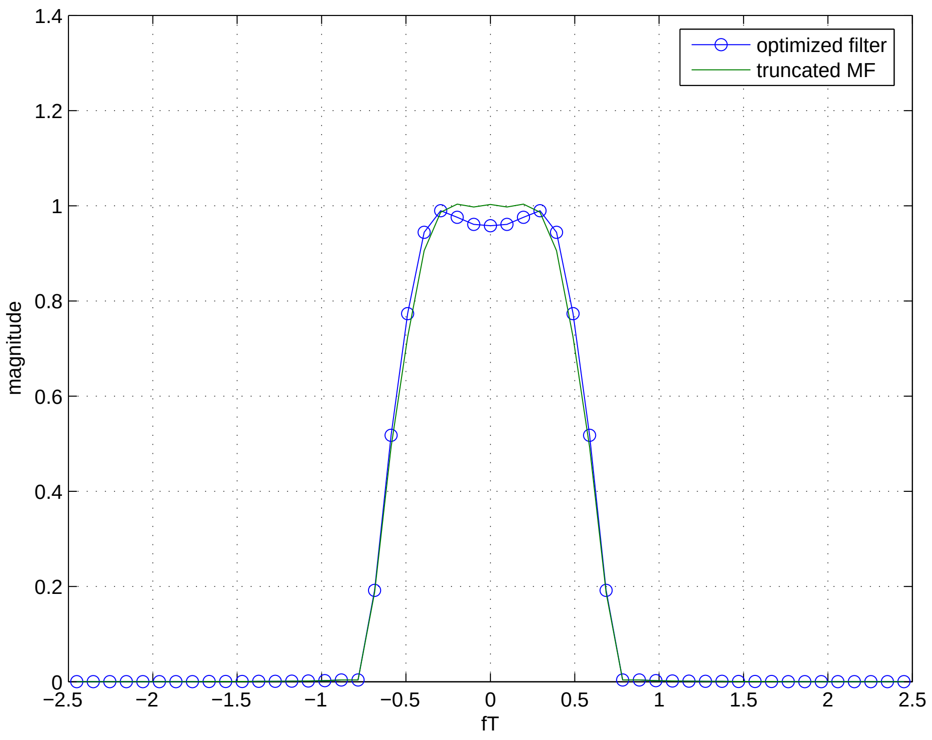


Figure 7.9: *Frequency response of original and optimized receive filter for $CIC(4, 3)$, $T_s = 1/11.1$, $N = 5$ and 51 filter taps*

To illustrate this, we consider a $CIC(4, 3)$ filter and $T_s/T_{min} = 1/11.1$, $N = 5$ for $T/T_{min} \in (1, 2)$. Fig. 7.9 shows the frequency response of a truncated matched filter (for a receive filter with 51 taps) as well as the filter resulting from the optimization criterion (7.25). Note that this optimization process is only useful in situations where the degradation is mainly due to the distortion of the useful signal (i.e., the passband droop) and not because of aliasing of HF components. This is not really a restriction as we may increase either R or L in $CIC(R, L)$ to suppress HF aliasing components.

7.4.6 Numerical Results

We will determine, as a function of $r = rem\,(2f_{IF}T_s)$, the maximum BER degradation (maximized over all considered symbol rates) assuming uncoded BPSK transmission at a reference BER of 10^{-3}, with $T_{max}/T_{min} = 2$, $B_{max} = 0.75/T_{min}$, $B_{AA} = 0.9/T_{min}$ and $T_{min}/T_s = 11.1$ (which satisfies (7.22)). We consider both the case of a discrete set of symbol rates (i.e., $T \in \{T_{min}, T_{max}\}$) and of a continuous set (i.e., $T \in [T_{min}, T_{max}]$). The lower bounds for N according to (7.20), (7.23) and (7.24) are given by $N = 3$, $N = 6$ and $N = 14$, respectively. We will consider three values of N ($N = 5$, $N = 12$ and $N = 16$). We will observe that in the case of baseband sampling, degradations are very low, both for the IP+MF and the MF+IP configuration. This confirms the results from [67]. As mentioned in section 7.4.5.2, the IP+MF:IF with ideal DAAF configuration and the MF+IP:IF configuration will lead to low degradations for a wide range of r. Around $r = 0$ and $r = 1$, HF components coincide with the useful signal, so that IF-sampling inevitably leads to severe degradations.

Case 1: N=16

Fig. 7.10 shows results for $N = 16$. When no digital AAF is present in the IP+MF:IF system, low degradations are visible for $r \approx r^* = 0.15$, $r \approx 1 - r^* = 0.85$ (with r^* defined in section 7.4.5.2) and for r around $1/2$. This is in accordance with the results from [75]. Observe that for r around $1/2$, the degradation is not very sensitive to the IF.

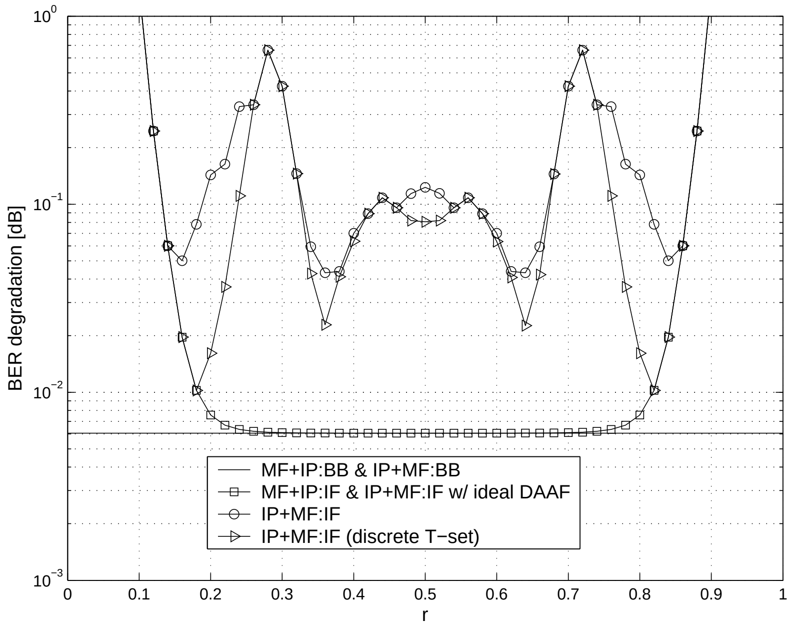


Figure 7.10: *BER degradation (at a reference BER of 10^{-3}) as a function of r maximized over the considered symbol rates for $N = 16$. For MF+IP the continuous-rates and discrete-rates cases coincide.*

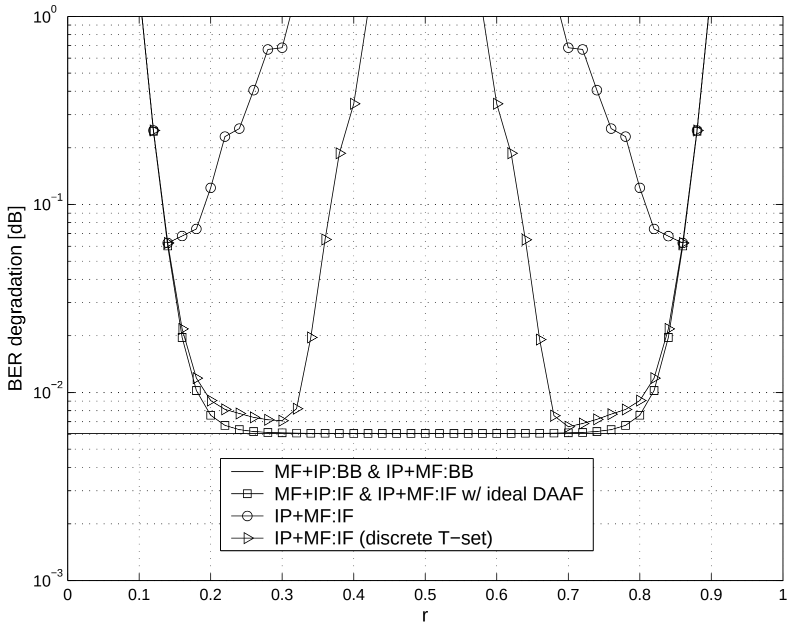


Figure 7.11: *BER degradation (at a reference BER of 10^{-3}) as a function of r maximized over the considered symbol rates for $N = 12$. For MF+IP the continuous-rates and discrete-rates cases coincide.*

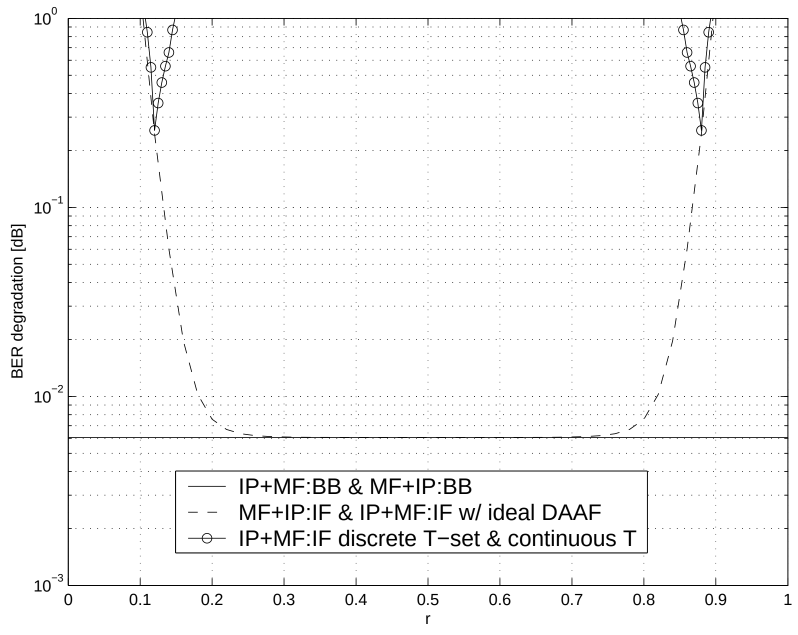


Figure 7.12: *BER degradation (at a reference BER of* 10^{-3}*) as a function of r for* $N = 5$ *without CIC filters*

Case 2: N=12

When we decrease N to 12, as shown in Fig. 7.11, we see that for IP+MF:IF without AAF, acceptable situations occur only for $r \approx r^*$ and $r \approx 1 - r^*$. Note the difference between the continuous symbol rate case and the discrete symbol rate case.

Case 3: N=5

Reducing N even further to $N = 5$, we obtain results shown in Fig. 7.12. Absence of a digital AAF in the IP+MF:IF receiver results in high degradations for almost every value of r, even when we only consider the discrete set of symbol rates. This figure clearly illustrates that applying a digital AAF can significantly reduce the BER degradation. In Fig. 7.13, we therefor consider efficient CIC filters with factor $R = 4$. From (7.8), we gather that HF components will undergo maximal attenuation when $r \in \{1/4, 1/2, 3/4\}$. This effect is clearly visible for the $CIC(4, 2)+IP+MF$-case: degradations of the IF-sampling receiver are equal to those of the BB-sampling receiver for those values of r. This means that for those values of r the degradation is now dominated by the distortion of the useful signal component (the passband droop). On the other hand, for values of r far enough away from $\{1/4, 1/2, 3/4\}$, IF-sampling results in an additional degradation compared to BB-sampling. This additional degradation is due to aliasing of HF signal and HF noise components. For those values of r, optimization of the receive filter will not significantly reduce the degradation. Note that for BB-sampling, the CIC filter has no useful purpose (it will only distort the useful signal), so the corresponding curves merely serve as a reference point (i.e., it shows to what extent the degradation is due to aliasing, rather than to passband droop). The $CIC(4, 3)$ leads to similar results, only now the degradation is dominated by distortion of the useful signal component for all $r \in [0.2, 0.8]$. Observe that even for BB-sampling the degradation is fairly high. This situation is well suited to the optimization method we proposed at the end of section 7.4.5.2.

As expected, optimization of the receive filter in the $CIC(4, 2)$ case yields the most gain when $r \in \{1/4, 1/2, 3/4\}$. In other r-regions, degradations have been reduced, but remain very sensitive to the value of r.

For $CIC(4, 3)$, optimization of the receive filter reduces the degradation, both for IF and BB-sampling. Again the degradation for IF-sampling remains roughly constant for $r \in [0.2, 0.8]$. Although the minimal degradation is higher compared to the optimized $CIC(4, 2)$ case, degradations remain low for a wide range of r, resulting in less sensitivity to the intermediate frequency.

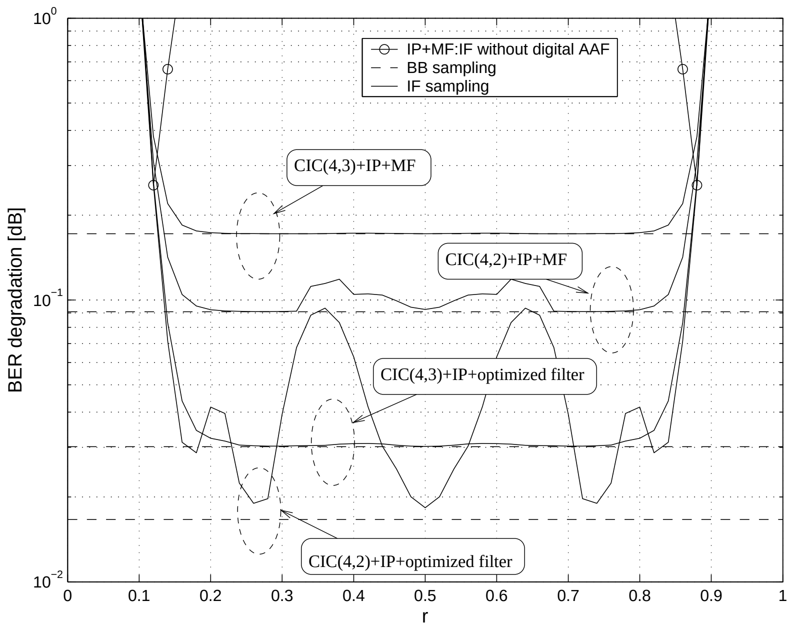


Figure 7.13: *BER degradation (at a reference BER of 10^{-3}) as a function of r for $N = 5$ with CIC filters with and without optimized receive filter.*

7.4.7 Complexity comparison

IP+MF

The design parameters $\{r, T_s, R, L, N\}$ each have a different impact on the complexity of the proposed receiver: the length of the matched filter is proportional to N, the complexity of the ADC and the digital AAF is related to T_s. For the CIC filter, the computational complexity is proportional to L and R. On the other hand, r has essentially no impact on receiver complexity. For practical reasons it is often preferred to keep f_{IF} as low as possible. Note that as T_s decreases, interpolation performance of the low-pass signal part improves: the main lobe of $P_I(f)$, the FT of $p_I(t)$, is broader and signal components centered around $f = k/T_s$, for $k \neq 0$, undergo more attenuation, leading to less aliasing due to baseband components.

MF+IP

For the configuration with matched filtering prior to interpolation, the parameter N has no meaning, because the ratio of the interpolator input and output rates is not usually an integer. The optional digital AAF has again a computational complexity related to T_s. Note that in this configuration the matched filter serves as a symbol rate dependent anti-aliasing filter. Hence the optional DAAF serves no useful purpose and can be omitted.

Comparison

We will compare two receiver configurations. On the one hand the CIC+IP+MF receiver and on the other hand the MF+IP receiver. We assume that $r = 0.27$, $T_{max}/T_{min} = 2$ and that the matched filter spans $K = 12$ symbol periods. From section 7.4.6, we see that for $T_{min}/T_s = 11.1$ and $N = 5$, the CIC($R = 4$,$L = 2$) filter + non-optimized matched filter leads to a BER degradation of roughly 0.1 dB. It can be verified that a MF+IP receiver with $T_{min}/T_s = 5.6$ yields about the same BER degradation. The computational complexity (measured in terms of the number of real-valued operations per symbol interval) of the MF+IP configuration can be approximated by $C_{MFIP} \approx 2K(T/T_s)^2$. On the other hand, the CIC+IP+MF configuration results in a computational complexity $C_{IPMF} \approx 2\,(LT/T_s + KN)$. When we substitute the considered system parameters, this yields $C_{MFIP} \in [753, 3010]$) and $C_{IPMF} \in [164, 209]$. In this case, the MF+IP configuration leads to a significantly larger processing time for the same degradation compared to the CIC+IP+non-optimized MF receiver.

As far as memory requirements are concerned, we remind that the IP+MF receiver requires storing only a single set of matched filter taps, whereas in the MF+IP receiver a different set of filter taps needs to be loaded for each value of the symbol rate.

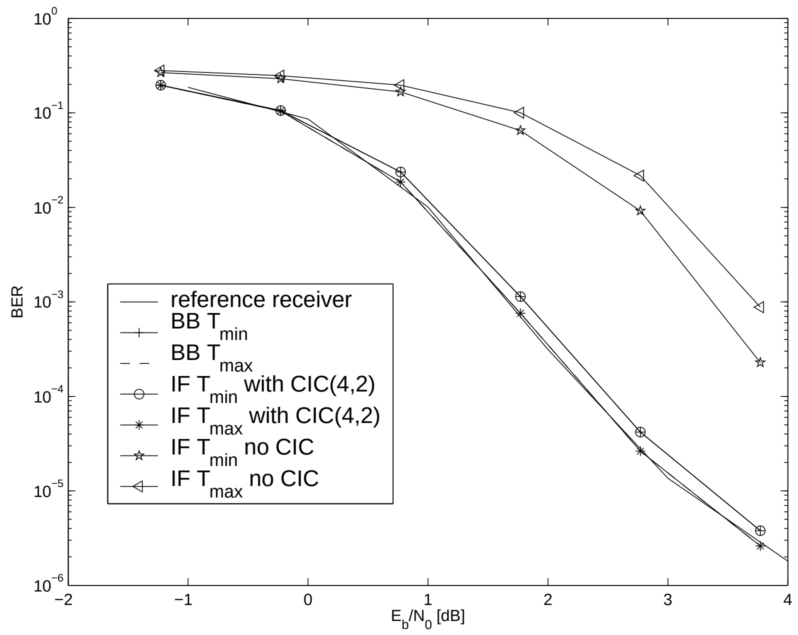


Figure 7.14: *BER performance of the proposed receiver for turbo-coded transmission for baseband (BB) sampling and IF-sampling.*

7.4.8 Remarks

Timing Recovery

Depending on system requirements, timing recovery could be performed either in feed-forward (FF) mode (e.g., for burst transmission) or in feed-back (FB) mode (e.g., for continuous transmission). Application of FF timing recovery to the receiver from Fig. 7.4 involves filtering the sequence $\{x(nT_s)\}$, feeding the resulting sequence to an even-symmetrical nonlinearity, and computing the timing estimate from the Fourier transform (evaluated at the frequency $1/T$) of the nonlinearity output [16]. From the timing estimate, the quantities μ_i and m_i are determined, and fed to the interpolator. In the case of FB timing recovery, the signal (at rate N/T) at the output of the receiver filter $H(\exp(j2\pi fT/N))$ is fed to a timing error detector [16]. Based on the timing error detector output and on the values of μ_i and m_i, the quantities μ_{i+1} and m_{i+1} are computed, and passed to the interpolator. A low-complexity FB timing estimator for IF-sampling receivers was proposed in [78], whereby the oversampling is embedded in the timing recovery circuit.

In a practical implementation, it may be more convenient to perform timing recovery and phase estimation (i.e., determining τ and θ) after matched filtering. It would therefore be natural to use two interpolators: the first one (before the matched filter) would take of sample rate conversion. Hence, it would not require any timing information: it simply inputs samples at rate $1/T_s$ and outputs samples at a (nominal) rate N/T, without any regard to the clock phase. A second interpolator (after the matched filter) would be used to reconstruct the signal at the correct timing instants. It replaces the decimator from Fig. 7.4.

IF-sampling issues

Although IF-sampling seems attractive, there are a number of issues that were omitted in the previous discussion. Current systems and circuits still suffer from several sources of degradation, mostly related to the ADC (such as sampling clock jitter) [79, 80]. These topics are beyond the scope of this dissertation.

7.5 Coded transmission

All the results above pertain to uncoded transmission. In [16, Chapter 7], it is mentioned that for coded transmission we expect very similar BER degradation performances as for uncoded transmission. To verify this, we have carried out computer simulations for a turbo code[4]. We consider two distinct symbol rates: $\{T_{min}, T_{max}\}$, with $T_{max} = 2T_{min}$,

[4]Parameters: the constituent convolutional coders are rate 1/2, recursive and systematic with octal generator $(21, 37)_8$, resulting in an overall rate equal to 1/3. The interleaver is pseudo-random and of length 111. BPSK transmission is assumed so that the block length equals $N_s = 333$.

while $B_{max} = 0.75/T_{min}$, $B_{AA} = 0.9/T_{min}$, $T_{min}/T_s = 11.1$ and $r = 0.5$. The matched filter operates at $N = 5$ samples per symbols (corresponding to Figs. 7.12-7.13).

BER performance results are shown in Fig. 7.14. As expected, the baseband sampling receivers result in a negligible degradation (less than 0.01 dB for $T = T_{max}$ and around 0.1 dB for $T = T_{min}$). In correspondence with the results from Fig. 7.12, the IF-sampling receivers result in very large degradations when no anti-aliasing filter is present. Applying the $CIC\,(4,2)$ filter reduces the degradation again to that of the baseband sampling receiver. We see that the design criteria remain valid for coded transmission.

7.6 Main Points

In this chapter, we have considered the problem of designing a low-complexity IF-sampling multi-rate receiver, suitable for multi-mode Software Radio. The proposed receiver combines IF-sampling with sample rate conversion before matched filtering. Aliasing is combatted by a special form of highly-efficient CIC filters. These filters can also be applied to perform channelization (i.e., the removal of unwanted signals from other users or services) by appropriate selection of the CIC design parameters.

Our design criterion was based on a frequency-domain interpretation of the signals, thus rendering it applicable to not only uncoded BPSK signaling, but also to coded systems, higher-order constellations and frequency-selective channels. This makes the four receiver structures roughly equivalent, assuming that the system is well-designed. Hence, any algorithm that operates using the matched filter outputs of one of these receivers can be applied with tolerable loss to the output of any of the remaining three. For the sake of clarity, we will consider only the baseband MF+IP receiver for the remainder of this dissertation. However, the reader should be aware that in practice we could/would use the IF-sampling IP+MF structure and obtain roughly the same results.

Only the first encoder is terminated.

Part III

Multi-mode receivers: estimation

Outline

Now that the problems of timing correction, equalization, mode adaptation and multi-rate receiver design are out of the way, we are ready to pick up where we left off in Chapter 2, namely the problem of estimation. The reader will have realized that in the second part of the dissertation, we have always assumed that the receiver has perfect knowledge of the channel impulse response, propagation delay, carrier phase and transmit mode. This is obviously a simplified view. In practice, some of these parameters will need to be estimated by the receiver. This will be the topic of this third part: estimation for multi-mode receivers. As conventional estimation algorithms fail due to the low SNR-low BER environment, we will develop so-called 'code-aided' (or 'code-aware') estimation algorithms that iterate between decoding and estimation.

- We start with some basic material pertaining to estimation algorithms in Chapter 8. This includes Maximum Likelihood (ML) and Maximum A Posteriori (MAP) estimation, and an interpretation based on factor graphs. We then consider a more practical class of estimation algorithms, namely the Expectation Maximization (EM) and Space Alternating Generalized Expectation Maximization (SAGE) algorithms.
- In Chapter 9 these techniques will be applied to our receiver resulting in a class of code-aided estimation algorithms that are able to exploit code properties during estimation in a systematic (as opposed to an ad-hoc) way.
- Unfortunately, the algorithms from Chapter 9 still suffer from two drawbacks. First of all, their computational complexity is excessive. Secondly, the proposed estimation algorithms require an initial estimate. When this initial estimate is unreliable the performance may be seriously degraded. These two problems will be addressed in Chapter 10, resulting in two novel estimation algorithms. One of these algorithms is particularly well suited to estimating discrete parameters.
- The problem of estimating discrete parameters has received little attention in the technical literature. In Chapter 11, we describe some relevant problems.
- Performance results are provided in Chapter 12. We will illustrate that the proposed algorithms can achieve impressive performances with little computational overhead. Additionally, the overhead related to training symbols can be reduced significantly, resulting in a gain in terms of power and bandwidth efficiency.

Chapter 8

Basic Principles: Estimation

8.1 Introduction

In this chapter, we will describe some general tools related to estimation. We start with the description of Maximum Likelihood (ML) and Maximum A Posteriori (MAP) estimation (section 8.3), including a factor graph interpretation (section 8.4). In most cases, these estimation techniques result in intractable algorithms. To overcome this, we present an iterative technique (known as the Expectation Maximization algorithm) that is able to perform ML/MAP estimation with a lower computational cost (section 8.5). As even the EM-algorithm cannot be applied to some multi-dimensional problems, we include a brief description of the Space-Alternating Generalized Expectation Maximization algorithm (the SAGE algorithm, section 8.6). This fairly abstract chapter ends with a more concrete example in section 8.7, to which we apply the different estimation techniques. Readers familiar with these topics can safely skip this chapter.

8.2 Problem formulation

The class of estimation problems we will consider can be formulated as follows:

Our goal is to estimate a parameter $\mathbf{d}_e$*, based on an observation* $\mathbf{r}$*. Furthermore, the observation* $\mathbf{r}$ *may depend on another unknown quantity* $\mathbf{d}_n$*. We refer to* $\mathbf{d}_n$ *as the nuisance parameter.*

The vectors $\mathbf{d}_e$ and $\mathbf{d}_n$ can take on values in a discrete or continuous domain, or a combination of both.

8.3 MAP and ML estimation

MAP and ML estimation are two possible techniques to compute estimates of a parameter [46]. MAP estimation exploits a priori information on the parameter. ML estimation is suited to problems where a priori information is missing, or when the parameter is non-random (i.e, unknown but deterministic).

8.3.1 Maximum A Posteriori estimation

When $\mathbf{a}$ is a random parameter, the MAP estimator maximizes the a posteriori probability:

$$\hat{\mathbf{d}}_{eMAP} = \arg\max_{\mathbf{d}_e} p\left(\mathbf{d}_e \left| \mathbf{r} \right.\right) \tag{8.1}$$

where[1]

$$\begin{aligned} p\left(\mathbf{d}_e \left|\mathbf{r}\right.\right) &= \sum_{\mathbf{d}_n} p\left(\mathbf{d}_e, \mathbf{d}_n \left|\mathbf{r}\right.\right) & (8.2)\\ &= \frac{p\left(\mathbf{d}_e\right)}{p\left(\mathbf{r}\right)} \sum_{\mathbf{d}_n} p\left(\mathbf{r} \left|\mathbf{d}_e, \mathbf{d}_n\right.\right) p\left(\mathbf{d}_n \left|\mathbf{d}_e\right.\right). & (8.3)\end{aligned}$$

In many cases, $\mathbf{d}_n$ will not depend on $\mathbf{d}_e$, so that $p\left(\mathbf{d}_n \left|\mathbf{d}_e\right.\right) = p\left(\mathbf{d}_n\right)$.

[1]Summations should be replaced by integrations, where applicable.

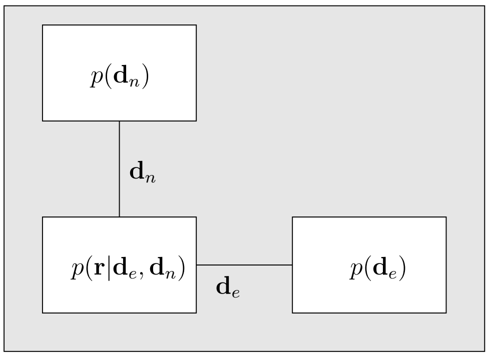


Figure 8.1: *Estimation on factor graphs: factor graph of* $p\left(\mathbf{d}_e, \mathbf{d}_n \left|\mathbf{r}\right.\right)$*. Each node can be further factorized, with the introduction of additional variables (edges).*

8.3.2 Maximum Likelihood estimation

When $\mathbf{d}_e$ is a non-random parameter or when $p\left(\mathbf{d}_e\right)$ is unknown, MAP estimation is not possible and maximum likelihood (ML) estimation is performed. In ML, we maximize the likelihood function w.r.t. $\mathbf{d}_e$:

$$\hat{\mathbf{d}}_{eML} = \arg\max_{\mathbf{d}_e} p\left(\mathbf{r}\left|\mathbf{d}_e\right.\right). \tag{8.4}$$

Observe that when $\mathbf{d}_e$ has a uniform a priori distribution over its entire domain, the MAP estimate reduces to the maximum likelihood (ML) estimate. Because of this relation, we will stick mainly to MAP estimation in this chapter. Particularizations to ML are obtained in a straightforward manner.

8.3.3 Two problems

ML and MAP estimation suffer from two problems:

1. First of all, the likelihood function $p\left(\mathbf{r}\left|\mathbf{d}_e\right.\right)$ and the a posteriori distribution $p\left(\mathbf{d}_e\left|\mathbf{r}\right.\right)$ are often very difficult to evaluate in practical scenarios, especially in the presence of nuisance parameters.

2. The second problem is related to the maximization: even when $p\left(\mathbf{r}\left|\mathbf{d}_e\right.\right)$ or $p\left(\mathbf{d}_e\left|\mathbf{r}\right.\right)$ can be evaluated in a reasonable amount of time, the maximization with respect to $\mathbf{d}_e$ can be very difficult: when $\mathbf{d}_e$ is multi-dimensional, maximization is often next to impossible, even through numerical methods.

8.4 Estimation on factor graphs

The solution to these problems seems obvious. Why not simply create a factor graph of the factorization of $p\left(\mathbf{d}_e, \mathbf{d}_n\left|\mathbf{r}\right.\right)$, perform the sum-product algorithm and that's it? Let us see how to proceed. First of all, we know that

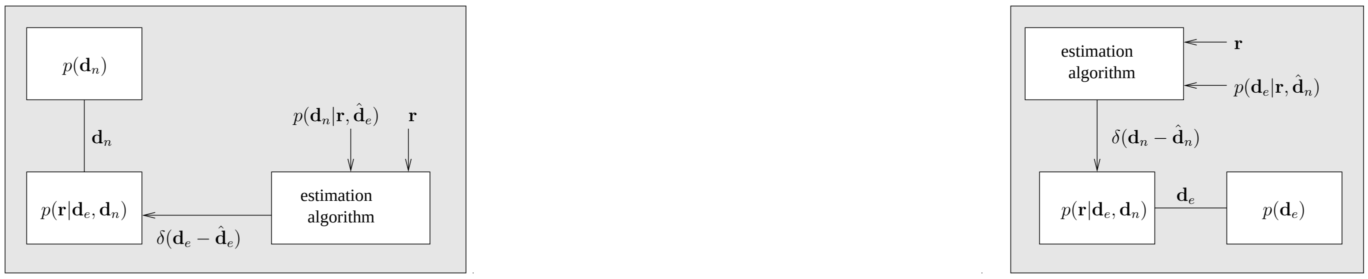
$$p\left(\mathbf{d}_e\left|\mathbf{r}\right.\right) = \sum_{\mathbf{d}_n} p\left(\mathbf{d}_e, \mathbf{d}_n\left|\mathbf{r}\right.\right) \tag{8.5}$$

and that

$$p\left(\mathbf{d}_e, \mathbf{d}_n\left|\mathbf{r}\right.\right) \quad \propto \quad p\left(\mathbf{r}\left|\mathbf{d}_e, \mathbf{d}_n\right.\right) p\left(\mathbf{d}_e\right) p\left(\mathbf{d}_n\right) \tag{8.6}$$

when $\mathbf{d}_e$ and $\mathbf{d}_n$ are independent. A factor graph of (8.6) is provided in Fig. 8.1.

Secondly, we will assume that $p\left(\mathbf{r}\left|\mathbf{d}_e, \mathbf{d}_n\right.\right)$, $p\left(\mathbf{d}_n\right)$ and $p\left(\mathbf{d}_e\right)$ have nice factorizations as function of the components of $\mathbf{d}_e$ (say, $d_{e,k}$) and $\mathbf{d}_n$ (say, $d_{n,l}$). Generally, this factorization will require the use of additional variables (see Chapter 3, section 3.3) and may create cycles in the graph. Hence, given some observation $\mathbf{r}$, we can apply the sum-product algorithm, yielding (approximations of) $p\left(d_{e,k}\left|\mathbf{r}\right.\right)$, $\forall k$ and $p\left(d_{n,l}\left|\mathbf{r}\right.\right)$, $\forall l$. So, applying the sum-product algorithm not only provides us with an estimate of $d_{e,k}$, but also of $d_{n,k}$! Problem solved. Or is it?



$p(\mathbf{d}_n)$
$\mathbf{d}_n$
$p(\mathbf{d}_n|\mathbf{r},\hat{\mathbf{d}}_e)$
$\mathbf{r}$
$p(\mathbf{r}|\mathbf{d}_e,\mathbf{d}_n)$
estimation algorithm
$\delta(\mathbf{d}_e-\hat{\mathbf{d}}_e)$
estimation algorithm
$\mathbf{r}$
$p(\mathbf{d}_e|\mathbf{r},\hat{\mathbf{d}}_n)$
$\delta(\mathbf{d}_n-\hat{\mathbf{d}}_n)$
$p(\mathbf{r}|\mathbf{d}_e,\mathbf{d}_n)$
$\mathbf{d}_e$
$p(\mathbf{d}_e)$

Figure 8.2: *Estimation through factor graphs: two pragmatic approaches*

8.4.1 Drawbacks

The sum-product algorithm in its pure form is not a popular tool for general estimation purposes. With good reason. In contrast to the discrete parameter estimation problems (e.g., estimating bits and symbols) from Part II of this dissertation, certain components $d_{e,k}$ or $d_{n,k}$ may take on values in a continuous domain, so that messages over the corresponding edges will be *probability density functions* (pdfs), rather than probability mass functions (which have convenient vector representations). So the question arises of how these pdfs should be represented. A list of samples? A single value? Parameters of some standard distributions? Furthermore, the summations in the sum-product algorithm are to be replaced with integrals. What should we do if we cannot obtain a closed-form solution for these integrals? Resort to numerical techniques? If so, which ones? How should the integration be related to the representation of the messages? Last of all, the factor graph of $p(\mathbf{r}\,|\mathbf{d}_e,\mathbf{d}_n)\,p(\mathbf{d}_e)\,p(\mathbf{d}_n)$ may contain many short cycles. These will degrade the performance of the sum-product algorithm and lead to unreliable marginals of $p(\mathbf{d}_e,\mathbf{d}_n\,|\mathbf{r})$, and thus to unreliable estimates. These reasons make estimation on factor graphs unattractive from an implementation-complexity point of view. This does not mean factor graphs are not used in estimation problems. We will describe some pragmatic approaches in the next section.

8.4.2 Pragmatic approach

The most common way to circumvent the problems described in the previous section is to break up the factor graph from Fig. 8.1, and to perform the sum-product algorithm on each of the disjoint sub-graphs. Messages between graphs may be of some special form (e.g., Dirac distributions).

Two extreme cases are depicted in Fig. 8.2. On the left, we estimate the parameter $\mathbf{d}_e$ through some existing algorithm (possibly the sum-product algorithm). The resulting estimate $\hat{\mathbf{d}}_e$ is provided to the $p(\mathbf{r}\,|\mathbf{d}_e,\mathbf{d}_n)$-block. The factor graph computes the marginals of $p\left(\mathbf{d}_n\,\middle|\hat{\mathbf{d}}_e,\mathbf{r}\right)$. These marginals can then be used by the estimation algorithm to improve the estimate of $\mathbf{d}_e$, leading to an iterative estimation procedure. Many estimation algorithms fall into this category, including the algorithms that will be described in the remainder of this chapter. On the other hand, we can interchange the role of $\mathbf{d}_e$ and $\mathbf{d}_n$, yielding the factor graph on the right of Fig. 8.2. A combination of these two extreme approaches is also possible. Just about any estimation algorithm one can conceive can be interpreted in this way.

8.5 The EM algorithm

As we have seen, straightforward application of the ML or MAP procedure is not possible in most estimation problems. Estimation on factor graph also has many drawbacks. The Expectation-Maximization (EM) algorithm is a technique that solves the MAP (or ML) problem in an iterative way [81, 82]. We will describe the MAP version of the EM algorithm (also known as the Bayesian EM algorithm [83]). The ML version is obtained by dropping terms related to the a priori distribution of $\mathbf{d}_e$.

8.5.1 Principle

Assume again that we want to estimate a parameter $\mathbf{d}_e$ from an observation $\mathbf{r}$. Suppose that, if we had access to another variable, say $\mathbf{d}_c$, the estimate of $\mathbf{d}_e$ could be easily computed, in a sense that $p(\mathbf{d}_e\,|\mathbf{r},\mathbf{d}_c)$ is easy to compute. The EM algorithm is an *iterative procedure* that exploits this variable $\mathbf{d}_c$ and at each iteration breaks down in two steps: the Expectation step (E-step) and the Maximization step (M-step).

Starting from an initial estimate $\hat{\mathbf{d}}_e^{(0)}$ of $\mathbf{d}_e$, the EM algorithm evaluates E- and M-steps successively and after each M-step produces a new estimate of $\mathbf{d}_e$. Hence, we obtain a sequence of estimates of $\mathbf{d}_e$: $\left[\hat{\mathbf{d}}_e^{(0)}, \hat{\mathbf{d}}_e^{(1)}, \hat{\mathbf{d}}_e^{(2)}, \ldots\right]$. When $\mathbf{d}_e$ is defined over a continuous domain, the EM algorithm will converge to a value $\hat{\mathbf{d}}_e^{(+\infty)}$, which is called a *solution* of the EM algorithm. Furthermore, the EM algorithm assures that the a posteriori probability (or the likelihood in case of ML estimation) of successive estimates is non-decreasing. Some issues related to the convergence of the EM algorithm are found in the Appendix of this chapter (section 8.9).

Complete data

We refer to $\mathbf{d}_c$ as the *complete data*. In order to qualify as valid complete data, we have the following sufficient condition on $\mathbf{d}_c$ [84]:

$$p\left(\mathbf{d}_c, \mathbf{r} \middle| \mathbf{d}_e\right) = p\left(\mathbf{r} \middle| \mathbf{d}_c\right) p\left(\mathbf{d}_c \middle| \mathbf{d}_e\right) \tag{8.7}$$

so that the observation can depend on $\mathbf{d}_e$ only through $\mathbf{d}_c$.

Initial estimate

An initial estimate $\hat{\mathbf{d}}_e^{(0)}$ of $\mathbf{d}_e$ is required to start the EM algorithm.

E-step

At iteration $i \geq 0$, the E-step is given by:

$$\begin{aligned} Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right) &= E_{\mathbf{d}_c}\left[\log p\left(\mathbf{d}_c, \mathbf{d}_e\right) \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right] && (8.8) \\ &= \log p\left(\mathbf{d}_e\right) + E_{\mathbf{d}_c}\left[\log p\left(\mathbf{d}_c \left|\mathbf{d}_e\right.\right) \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right] && (8.9) \\ &= \log p\left(\mathbf{d}_e\right) + \int \log p\left(\mathbf{d}_c \left|\mathbf{d}_e\right.\right) p\left(\mathbf{d}_c \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right) d\mathbf{d}_c. && (8.10) \end{aligned}$$

M-step

Following the E-step, the M-step is given by:

$$\hat{\mathbf{d}}_e^{(i+1)} = \arg\max_{\mathbf{d}_e} \left\{ Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right)\right\}. \tag{8.11}$$

8.5.2 Complete data and missing data

In many technical papers, $\mathbf{d}_c = [\mathbf{r}, \mathbf{d}_m]$, where $\mathbf{d}_m$ is referred to as the *missing* (or unobserved) data. In that case, $\mathbf{d}_c$ is always a valid complete data. Often, $\mathbf{d}_m$ will be independent of $\mathbf{d}_e$, so that the E-step becomes:

$$\begin{aligned} & Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right) \\ &= \log p\left(\mathbf{d}_e\right) + \int \log p\left(\mathbf{r}, \mathbf{d}_m \left|\mathbf{d}_e\right.\right) p\left(\mathbf{d}_m \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right) d\mathbf{d}_m && (8.12) \\ &\propto \log p\left(\mathbf{d}_e\right) + \int \log p\left(\mathbf{r} \left|\mathbf{d}_m, \mathbf{d}_e\right.\right) p\left(\mathbf{d}_m \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right) d\mathbf{d}_m. && (8.13) \end{aligned}$$

8.5.3 EM-based estimation using factor graphs

Although at this point it seems rather obvious, the EM algorithm can be implemented on a suitable factor graph. As was mentioned in [85], the a posteriori distribution $p\left(\mathbf{d}_c \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$ required during the E-step (8.8) can be obtained by defining a factor graph as in Fig. 8.2 (left part). The M-step is similarly obtained by defining a suitable factor graph where the sum and product operations are replaced with the maximization and sum operations, respectively[2], leading to the max-sum algorithm.

[2]Observe that $(\mathbb{R}, \max, +)$ is a commutative semi-ring. See Chapter 3, section 3.2.

8.6 The SAGE algorithm

8.6.1 Principle

The EM algorithm has the ability to circumvent the problem of ML and MAP estimation related to the nuisance parameter. However, the EM algorithm still suffers from the other ML/MAP problem: the optimization of a multi-dimensional function (i.e., the M-step). Additionally, there is the problem of the convergence rate: in order to make the M-step tractable, complete data spaces have to be defined that are sufficiently informative[3]: making the complete data space more informative (e.g., by adding mode parameters to it) results in easier maximization, but, at the same time leads to a reduction in the asymptotic convergence rate [81,84].

A solution to both of these problems was proposed in [84], known as the Space Alternating Generalized Expectation Maximization (or SAGE) algorithm, and can be seen as an extension of the EM algorithm. Although the SAGE algorithm was originally proposed in the context of image processing, it is, very much like the EM algorithm, applicable to general estimation problems. The SAGE algorithm operates in a way very similar to the EM algorithm. Rather than defining complete data, SAGE uses so-called *hidden data*. The E-step and M-step are also slightly modified.

As with the EM algorithm, convergence of the SAGE algorithm is assured in a sense that the a posteriori probability/likelihood of successive estimates is non-decreasing. The convergence proof is similar to the proof of the EM algorithm and can be found in [84].

Hidden Data

We break up $\mathbf{d}_e = [\mathbf{d}_{e,1}, \mathbf{d}_{e,2}, \ldots, \mathbf{d}_{e,M}]$, where, with a slight abuse of notation, $\mathbf{d}_{e,k} \subseteq \mathbf{d}_e$, and $\mathbf{d}_{e,\bar{k}} \doteq \mathbf{d}_e \setminus \mathbf{d}_{e,k}$. Each subset $\mathbf{d}_{e,k}$ has an associated so-called *hidden data* $\mathbf{d}_{h,k}$ such that $\mathbf{d}_{h,k}$ is a *complete data* for $\mathbf{d}_{e,k}$ when $\mathbf{d}_{e,\bar{k}}$ is known. Mathematically, this translates to (see Eq. (8.7))

$$p\left(\mathbf{d}_{h,k}, \mathbf{r} \middle| \mathbf{d}_e\right) = p\left(\mathbf{r} \middle| \mathbf{d}_{h,k}, \mathbf{d}_{e,\bar{k}}\right) p\left(\mathbf{d}_{h,k} \middle| \mathbf{d}_e\right). \tag{8.14}$$

When $M = 1$, the SAGE algorithm reduces to the standard EM algorithm.

Initial estimate

An initial estimate $\hat{\mathbf{d}}_e^{(0)}$ of $\mathbf{d}_e$ is again required to start the SAGE algorithm.

E-step

We now execute the EM algorithm, with the following alteration: at each iteration we only update a single $\mathbf{d}_{e,k}$, while the estimate of $\mathbf{d}_{e,\bar{k}}$ is left unchanged. When updating $\mathbf{d}_{e,k}$ we use the hidden data space $\mathbf{d}_{h,k}$. The E-step (8.8) thus becomes, at iteration $i \geq 0$:

$$\begin{aligned} & Q\left(\mathbf{d}_{e,k} \middle| \hat{\mathbf{d}}_e^{(i)}\right) \\ & = E_{\mathbf{d}_{h,k}}\left[\log p\left(\mathbf{d}_{h,k}, \mathbf{d}_{e,k}, \hat{\mathbf{d}}_{e,\bar{k}}^{(i)}\right) \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right] && (8.15) \\ & = \log p\left(\mathbf{d}_{e,k}, \hat{\mathbf{d}}_{e,\bar{k}}^{(i)}\right) + E_{\mathbf{d}_{h,k}}\left[\log p\left(\mathbf{d}_{h,k} \middle| \mathbf{d}_{e,k}, \hat{\mathbf{d}}_{e,\bar{k}}^{(i)}\right) \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right] && (8.16) \\ & = \log p\left(\mathbf{d}_{e,k}, \hat{\mathbf{d}}_{e,\bar{k}}^{(i)}\right) + \int \log p\left(\mathbf{d}_{h,k} \middle| \mathbf{d}_{e,k}, \hat{\mathbf{d}}_{e,\bar{k}}^{(i)}\right) p\left(\mathbf{d}_{h,k} \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right) d\mathbf{d}_{h,k} && (8.17) \end{aligned}$$

which is now a function of $\mathbf{d}_{e,k}$ only.

M-step

The M-step now becomes:

$$\left\{ \begin{array}{lcl} \hat{\mathbf{d}}_{e,k}^{(i+1)} & = & \arg\max_{\mathbf{d}_{e,k}} Q\left(\mathbf{d}_{e,k} \middle| \hat{\mathbf{d}}_e^{(i)}\right) \\ \hat{\mathbf{d}}_{e,\bar{k}}^{(i+1)} & = & \hat{\mathbf{d}}_{e,\bar{k}}^{(i)} \end{array} \right. \tag{8.18}$$

[3]It has been observed that in the selection of the complete data, one can trade convergence rate for complexity through the 'informativeness' of the complete data. Information should here be understood in the sense of the Fisher Information Matrix (FIM): when more information is added to make up the complete data $\mathbf{d}_c$, the EM algorithm becomes easier to solve (both the E-step and the M-step), but the convergence rate decreases. For proofs, see [81].

The main differences between the SAGE and the EM algorithm are:

- SAGE updates parameter subsets rather than the entire parameter set: this makes the maximizations tractable;
- SAGE uses hidden data spaces that are particular to the parameter subset. By creating hidden data spaces that are sufficiently informative only for that subset, overall asymptotic convergence can be increased.

8.7 An example

The SAGE algorithm is commonly applied when the M-step in the EM algorithm is difficult or when the convergence rate of the EM algorithm is poor. A popular example is that of signal decomposition. To allow the reader to get some familiarity with the ML, EM and SAGE methods, we will consider this example in some detail. The problem description is based on [86,87]. This example will turn out to be very relevant for our particular problem (estimation of channel parameters) as will become apparent in Chapter 9, section 9.3.

8.7.1 Problem formulation

Suppose we have an observation, $\mathbf{r}$, that is the sum of L signals, $\mathbf{s}_0, \ldots, \mathbf{s}_{L-1}$, corrupted with AWGN. Each of the signals $\mathbf{s}_k$ depends on a parameter $\mathbf{b}_k$ and a common nuisance parameter $\mathbf{a}$. Introducing $\mathbf{b} = \left[\mathbf{b}_0^T, \ldots, \mathbf{b}_{L-1}^T\right]^T$, we have

$$\begin{aligned}
\mathbf{r} &= \sum_{k=0}^{L-1} \mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right) + \mathbf{n} && (8.19)\\
&\doteq \mathbf{s}\left(\mathbf{a}, \mathbf{b}\right) + \mathbf{n} && (8.20)
\end{aligned}$$

where $E\left[\mathbf{n}\mathbf{n}^H\right] = 2\sigma^2\mathbf{I}$. This implies

$$\log p\left(\mathbf{r}\left|\mathbf{b}, \mathbf{a}\right.\right) \propto -\frac{1}{2\sigma^2}\left\|\mathbf{r} - \mathbf{s}\right\|^2. \tag{8.21}$$

We will work under the following assumptions:

- $\mathbf{b}$ and $\mathbf{a}$ are a priori independent;
- $p\left(\mathbf{a}\right)$ is known;
- we have available or can easily compute[4] $p\left(\mathbf{a}\left|\mathbf{r}, \mathbf{b}\right.\right)$ for any $\mathbf{b}$.

Suppose we want to obtain an ML estimate of $\mathbf{d}_e = \mathbf{b}$ in the presence of a nuisance parameter $\mathbf{d}_n = \mathbf{a}$. We will apply three techniques to estimate $\mathbf{b}$:

1. The ML-technique
2. The EM algorithm
3. The SAGE algorithms

8.7.2 ML estimation

The ML estimate is given by (8.4):

$$\hat{\mathbf{b}}_{ML} = \arg\max_{\mathbf{b}} p\left(\mathbf{r}\left|\mathbf{b}\right.\right) \tag{8.22}$$

where

$$p\left(\mathbf{r}\left|\mathbf{b}\right.\right) = \int p\left(\mathbf{r}\left|\mathbf{b}, \mathbf{a}\right.\right) p\left(\mathbf{a}\right) d\mathbf{a}. \tag{8.23}$$

Unfortunately, (8.23) is generally impossible to evaluate in a closed form. Furthermore, the maximization (8.22) is hard when L is large or when each $\mathbf{b}_k$ contains many components. These two problems make ML estimation impossible to use in practice.

[4]Although this assumption may require a leap of faith on the part of the reader, a motivation will be provided in later chapters.

8.7.3 EM estimation

Let us try the EM approach.

Complete data definition

We consider $\mathbf{a}$ as missing data so that $\mathbf{d}_c = [\mathbf{r}, \mathbf{a}]$.

E-step

Applying the EM algorithm with complete data $\mathbf{d}_m = [\mathbf{r}, \mathbf{a}]$, the E-step (8.13) becomes

$$\begin{aligned}
Q\left(\mathbf{b}\left|\hat{\mathbf{b}}^{(i)}\right.\right) &\propto \int \log p\left(\mathbf{r}\left|\mathbf{a}, \mathbf{b}\right.\right) p\left(\mathbf{a}\left|\mathbf{r}, \hat{\mathbf{b}}^{(i)}\right.\right) d\mathbf{a} && (8.24)\\
&\propto -\int \left\|\mathbf{r} - \mathbf{s}\left(\mathbf{a}, \mathbf{b}\right)\right\|^2 p\left(\mathbf{a}\left|\mathbf{r}, \hat{\mathbf{b}}^{(i)}\right.\right) d\mathbf{a} && (8.25)\\
&\propto -E_{\mathbf{a}}\left[\left\|\mathbf{s}\left(\mathbf{a}, \mathbf{b}\right)\right\|^2 \middle| \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right] && (8.26)\\
&+ 2\Re\left\{\mathbf{r}^H \sum_{k=0}^{L-1} E_{\mathbf{a}}\left[\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\middle| \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right]\right\}
\end{aligned}$$

Since $p\left(\mathbf{a}\left|\mathbf{r}, \hat{\mathbf{b}}^{(i)}\right.\right)$ is assumed to be available, we can in principle evaluate $Q\left(\mathbf{b}\left|\hat{\mathbf{b}}^{(i)}\right.\right)$. Hence, the first problem related with ML estimation is now circumvented.

M-step

Since $Q\left(\mathbf{b}\left|\hat{\mathbf{b}}^{(i)}\right.\right)$ is a function of $\mathbf{b}$, maximizing (8.26) w.r.t. $\mathbf{b}$ is still L-dimensional and thus still very hard.

8.7.4 SAGE estimation

We can also apply the SAGE algorithm with hidden data[5] for $\mathbf{b}_k$, $\mathbf{d}_{h,k} = [\mathbf{a}, \mathbf{r}]$. In this case the E-step is given by

$$\begin{aligned}
& Q\left(\mathbf{b}_k \middle| \hat{\mathbf{b}}^{(i)}\right) && (8.27)\\
& \propto \int \log p\left(\mathbf{r}\middle| \mathbf{a}, \mathbf{b}_k, \hat{\mathbf{b}}_{\bar{k}}^{(i)}\right) p\left(\mathbf{a}\left|\mathbf{r}, \hat{\mathbf{b}}^{(i)}\right.\right) d\mathbf{a}\\
& \propto -\int \left\|\left(\mathbf{r} - \sum_{l\neq k} \mathbf{s}_l\left(\mathbf{a}, \hat{\mathbf{b}}_l^{(i)}\right)\right) - \mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\right\|^2 p\left(\mathbf{a}\left|\mathbf{r}, \hat{\mathbf{b}}^{(i)}\right.\right) d\mathbf{a}\\
& \propto -E_{\mathbf{a}}\left[\left\|\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\right\|^2 \middle| \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right] + 2\Re\left\{E_{\mathbf{a}}\left[\hat{\mathbf{z}}_k^H\left(\mathbf{a}, \hat{\mathbf{b}}_{\bar{k}}^{(i)}\right) \mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\middle| \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right]\right\}
\end{aligned}$$

where

$$\hat{\mathbf{z}}_k\left(\mathbf{a}, \hat{\mathbf{b}}_{\bar{k}}^{(i)}\right) = \mathbf{r} - \sum_{l\neq k} \mathbf{s}_k\left(\mathbf{a}, \hat{\mathbf{b}}_l^{(i)}\right). \qquad (8.28)$$

Note that the SAGE algorithm has a rather nice interpretation: to estimate $\mathbf{b}_k$, we deduct from $\mathbf{r}$ all interfering signals (i.e., $\sum_{l\neq k} \mathbf{s}_k\left(\mathbf{a}, \hat{\mathbf{b}}_l^{(i)}\right)$). The remaining signal is then treated as the sum of $\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)$ and an AWGN term. This removal of interference bears much resemblance to Serial Interference Cancellation (SIC), a common technique in multi-user detection.

8.8 Main points

In this chapter we have given an overview of several general-purpose estimation techniques. We started with Maximum likelihood (ML) and Maximum A Posteriori (MAP) estimation. Both ML and MAP estimation suffer from

[5]Note that we select the hidden data to be the same for all parameter subsets.

several problems: they require the optimization of functions of many parameters. On top of that, the functions themselves are generally very hard to compute.

The Expectation-Maximization algorithm can be used to iteratively solve MAP and ML problems at a lower computational cost. Finally, we have briefly described the Space-Alternating Generalized EM algorithm (SAGE) whereby at each iteration only a subset of parameters is updated. This last technique is especially well-suited to multi-dimensional problems.

8.9 Appendix: Convergence of the EM algorithm

We will show that the MAP (or ML) estimate is a solution of the EM algorithm and that for any initial estimate, $\hat{\mathbf{d}}_e^{(0)}$, subsequent estimates will have an increasing a posteriori probability (or likelihood, for ML). We focus on MAP estimation.

Theorem 8.9.1. *The a posteriori probability of two subsequent estimates in the EM algorithm is non-decreasing:*

$$\log p\left(\hat{\mathbf{d}}_e^{(i+1)}\left|\mathbf{r}\right.\right) \geq \log p\left(\hat{\mathbf{d}}_e^{(i)}\left|\mathbf{r}\right.\right). \tag{8.29}$$

Proof. Due to the definition of the M-step (8.11), two subsequent estimates necessarily satisfy:

$$Q\left(\hat{\mathbf{d}}_e^{(i+1)}\middle|\,\hat{\mathbf{d}}_e^{(i)}\right) \quad \geq \quad Q\left(\hat{\mathbf{d}}_e^{(i)}\middle|\,\hat{\mathbf{d}}_e^{(i)}\right). \tag{8.30}$$

Furthermore, taking into account (8.7), it is easily verified that

$$\begin{aligned} Q\left(\mathbf{d}_e\middle|\,\hat{\mathbf{d}}_e^{(i)}\right) &= \log p\left(\mathbf{r}\right) - \int \log p\left(\mathbf{r}\middle|\,\mathbf{d}_c\right) p\left(\mathbf{d}_c\middle|\hat{\mathbf{d}}_e^{(i)},\mathbf{r}\right) d\mathbf{d}_c \\ &+ \log p\left(\mathbf{d}_e\left|\mathbf{r}\right.\right) + \int \log p\left(\mathbf{d}_c\middle|\,\mathbf{d}_e,\mathbf{r}\right) p\left(\mathbf{d}_c\middle|\hat{\mathbf{d}}_e^{(i)},\mathbf{r}\right) d\mathbf{d}_c. \end{aligned} \tag{8.31}$$

Hence, after some rearranging of (8.30), we obtain

$$\begin{aligned} &\log p\left(\hat{\mathbf{d}}_e^{(i+1)}\left|\mathbf{r}\right.\right) - \log p\left(\hat{\mathbf{d}}_e^{(i)}\left|\mathbf{r}\right.\right) \\ &\geq \int \log \frac{p\left(\mathbf{d}_c\middle|\,\hat{\mathbf{d}}_e^{(i)},\mathbf{r}\right)}{p\left(\mathbf{d}_c\middle|\,\hat{\mathbf{d}}_e^{(i+1)},\mathbf{r}\right)} p\left(\mathbf{d}_c\middle|\hat{\mathbf{d}}_e^{(i)},\mathbf{r}\right) d\mathbf{d}_c. \end{aligned} \tag{8.32}$$

The right-hand-side in (8.32) is nothing more than the Kullback-Leibler Distance (KLD) between two distributions [88]. Since the KLD is non-negative, this proves (8.29).

□

Theorem 8.9.2. *The MAP estimate is also a solution of the EM algorithm:*

$$\hat{\mathbf{d}}_{e,MAP} = \arg\max_{\mathbf{d}_e} Q\left(\mathbf{d}_e\middle|\,\hat{\mathbf{d}}_{e,MAP}\right) \tag{8.33}$$

Proof. This follows immediately from the previous Theorem and the fact that $\log p\left(\hat{\mathbf{d}}_{e,MAP}\left|\mathbf{r}\right.\right) \geq \log p\left(\mathbf{d}_e\left|\mathbf{r}\right.\right)$ for all $\mathbf{d}_e$.

□

The convergence behavior of the EM algorithm should be interpreted as depicted in Fig. 8.3: for any value of $\mathbf{d}_e$, there are multiple solutions of the EM algorithm in the space where $\mathbf{d}_e$ resides. Around each of these solutions lies a domain of attraction: when the initial estimate is within one of those domains, the EM algorithms converges (with high probability[6]) to the corresponding solution. One of these solutions is the ML (or MAP) estimate. We will name the domain of attraction around the ML (MAP) estimate, the acquisition region.

[6]The boundaries of the domains of attraction are not 'hard' boundaries.

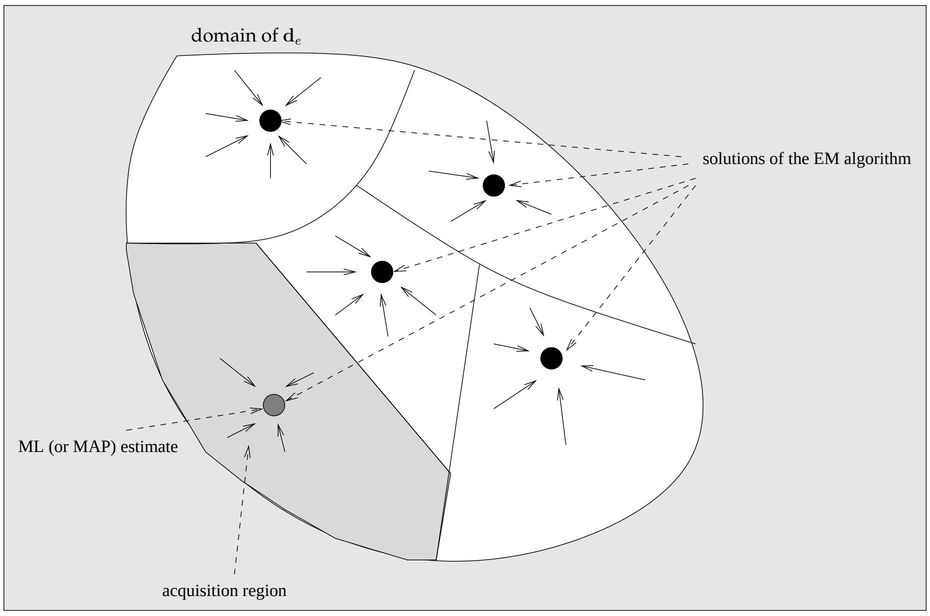


Figure 8.3: *Convergence behavior of the EM algorithm*

Chapter 9

Code-aided estimation: the big picture

9.1 Introduction

Now that we have a firm grasp of the principles of estimation theory, we will apply Maximum Likelihood (ML) techniques to the system from Chapter 2. We will consider three classes of algorithms to estimate the channel impulse response and/or synchronization parameters. First of all, we describe a class of 'conventional' estimation algorithms. These can be divided in Data-Aided algorithms (which exploit the knowlegde of known data symbols in the burst - known as pilot symbols or training symbols) and Non-Data-Aided (NDA) algorithms (which are based on statistical properties of the incoming signal). Secondly, we will derive algorithms that are able to exploit code properties during estimation through the EM and SAGE principles. We then briefly discuss a third class of algorithms that perform joint decoding and estimation by extending the factor graph at the receiver to include the uncertainty w.r.t. the channel parameters.

Many other such 'code-aided' algorithms have been proposed in the last decade or so. Still, most of them are either closely related to EM- or factor-graph-based estimators, or are more of an ad-hoc nature. As the latter class of algorithms cannot be applied to a wide range of problems and are inherently sub-optimal, we will stick to more systematic techniques in this dissertation.

In Fig. 9.1, the receiver from Fig. 6.1 is repeated with one additional block: the estimator. The estimator has as inputs the received signal, exact knowledge regarding pilot symbols, and, for a code-aided estimator, some kind of information from the detector. The output of the estimator is, depending on the set-up, an estimate of the channel impulse response, or of the channel parameters, or of the synchronization parameters, or of the transmit mode.

This chapter deals mainly with describing how information should flow from the detector to the estimator. In a first phase, we will assume that the transmit mode is known at the receiver.

9.2 Conventional estimation techniques

We will start with estimation techniques for frequency-selective channels. Since a frequency-flat channel is merely a special case of a frequency-selective channel, estimation algorithms for the latter channel can be applied unmodified to the former. However, we will point out that more suitable algorithms exist, tailored especially to the frequency-flat channel. Tailoring algorithms to suit a certain estimation problem will be a common theme in this and the following chapters: although the techniques we have presented (ML, MAP, EM and SAGE) are very general, care needs to be taken when they are applied to specific estimation problems. Sometimes a certain algorithm will be overkill for a certain estimation problem. Also, some parameters require different estimation strategies than others.

9.2.1 Frequency-selective channel

We start from the original model, given in Eq. (2.12) in Chapter 2. In frequency-selective channels, channel estimation is usually performed by exploiting pilot symbols [16]. The signal at the input of the receiver is of the form

$$\begin{aligned} r_{BB}(t) &= \sum_{n=0}^{N_s-1} a_n h(t-nT) + w(t) && (9.1) \\ &= \sum_{l=0}^{L-1} \alpha_l \sum_{n=0}^{N_s-1} a_n p(t-nT-\tau_l) + w(t) && (9.2) \end{aligned}$$

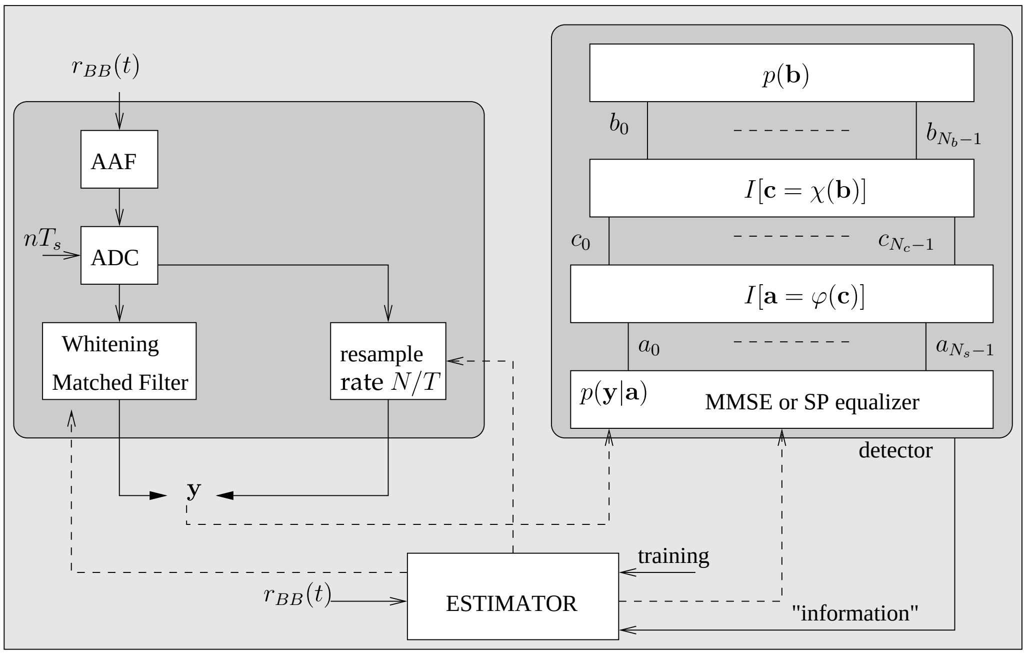


Figure 9.1: *Receiver with three main blocks: conversion of* $r_{BB}(t)$ *to* $\mathbf{y}$*, factor graph detector and estimator.*

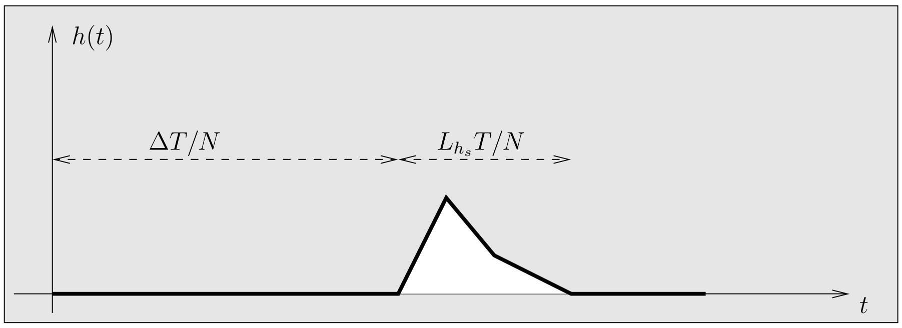


Figure 9.2: *Channel estimation: unstructured approach*

where a_n is the n-th data symbol, $n(t)$ is a complex white Gaussian noise process with PSD σ^2 per real dimension. The first P symbols are pilot symbols (i.e., known to the receiver), while the remaining $N_s - P$ are unknown coded symbols, so that $\mathbf{a} = [\mathbf{a}_P^T \, \mathbf{a}_C^T]^T$. The signal is band-limited and sampled at a rate $1/T_s = N/T$, yielding samples $\{r(kT_s)\}$. Considering the results from Chapter 5, we need to determine the overall impulse response

$$h(t) = \sum_{l=0}^{L-1} \alpha_l p(t - \tau_l) \tag{9.3}$$

We now have the choice to estimate $h(t)$ either directly or through the parameters ($\{\tau_k, \alpha_k\}_{k=0,\ldots,L-1}$) of the underlying channel model. These approaches lead to what is known as *unstructured* and *structured* estimators[1], respectively. Once the channel impulse response is known, the equalization techniques from Chapter 5 can be applied.

9.2.1.1 Method 1: the unstructured estimator

We refer back to the three observation models from Chapter 5, section 5.3: the matched filter receiver, the whitening matched filter receiver and the oversampling receiver. Note that the former two observation models require knowledge of the channel impulse response (CIR) to generate the observation, so it makes no sense to estimate the CIR from them. Hence, we start from an oversampled version of the signal. In Chapter 5, we have seen that when we start from an oversampled (at rate N/T) version of the received signal $r_{BB}(t)$, we obtain the following observation (see Eq. (5.21)):

$$\mathbf{r} = \mathbf{A}\mathbf{h} + \mathbf{w} \tag{9.4}$$

where $\mathbf{h}$ is the sampled channel impulse response of $h(t)$ and $\mathbf{A}$ is a matrix containing the data symbols $\mathbf{a}$ (see section 5.3.3 in Chapter 5). Generally, $h(t)$ will be of the form shown in Fig. 9.2:

$$h(t) = h_s\left(t - \Delta\frac{T}{N}\right) \tag{9.5}$$

where $h_s(t)$ is the part of $h(t)$ that takes on significant values (say, over a time interval $t = -L_l T, -L_l T + T/N, \ldots, +L_r T$). This leads to two estimation approaches:

Approach A Jointly estimating[2] $\Delta \in [0, \Delta_{max}]$ and the sampled version of $h_s(t)$. Denoting by L_{h_s} the duration of $h_s(t)$ expressed in number of symbol durations, this leads to the following observation model:

$$\begin{aligned} \mathbf{r} &= \begin{bmatrix} \mathbf{0}_\Delta \\ \mathbf{A}_D \\ \mathbf{0}_{(\Delta_{max}-\Delta)} \end{bmatrix} \begin{pmatrix} h_s(-L_l T) \\ \vdots \\ h_s(L_r T) \end{pmatrix} + \mathbf{w} & (9.6) \\ &= \mathbf{A}_\Delta \mathbf{h} + \mathbf{w} & (9.7) \end{aligned}$$

where $\mathbf{0}_x$ is an $x \times (L_{h_s}N)$ matrix of all zeros, and $\mathbf{A}_D$ is the matrix of data symbols, defined in section 5.3.3 in Chapter 5.

[1]In the technical literature, one may also find the names *parametric* (instead of structured) and *non-parametric* (instead of unstructured) estimators.

[2]Assume we know a priori that Δ lies in a known interval and have a rough idea of the duration of $h_s(t)$.

Approach B Estimating the sampled version of the entire channel impulse response $h(t)$, in which case we replace in (9.6) Δ_{max} and Δ with 0 and $\mathbf{h}$ with $[h(-L_l T_s), \ldots, h(L_r T_s + \Delta_{max} T/N)]^T$.

As Approach B can be cast within the framework of Approach A, we only focus on Approach A.

A data-aided estimation algorithm can be derived as follows: the receiver performs estimation under the assumption that unknown data symbols are all zero. We then get the following observation model:

$$\begin{aligned} \mathbf{r} &= \begin{bmatrix} \mathbf{0}_{\Delta} \\ \mathbf{A}_P \\ \mathbf{0}_{(\Delta_{max}-\Delta)} \end{bmatrix} \mathbf{h} + \mathbf{w} && (9.8) \\ &= \mathbf{A}_{\Delta,P}\mathbf{h} + \mathbf{w} && (9.9) \end{aligned}$$

where $\mathbf{A}_P$ is obtained by replacing in $\mathbf{A}_D$ all unknown coded data symbols with zeros. Applying the ML principle to obtain a data-aided estimate of $\mathbf{h}$ and Δ now yields

$$\left[\hat{\mathbf{h}}, \hat{\Delta}\right] = \arg\max_{\mathbf{h},\Delta} \log p(\mathbf{r}\,|\mathbf{h}, \Delta). \quad (9.10)$$

Since

$$\log p(\mathbf{r}\,|\mathbf{h}, \Delta) \propto -\mathbf{h}^H \mathbf{A}_P^H \mathbf{A}_P \mathbf{h} + 2\Re\left\{\mathbf{r}^H \mathbf{A}_{\Delta,P}\mathbf{h}\right\}, \quad (9.11)$$

(9.10) is solved by

$$\hat{\Delta} = \arg\max_{\Delta}\left\{\Re\left(\mathbf{r}^H \mathbf{A}_{\Delta,P}\left(\mathbf{A}_P^H\mathbf{A}_P\right)^{-1}\mathbf{A}_{\Delta,P}^H\mathbf{r}\right)\right\} \quad (9.12)$$

and

$$\hat{\mathbf{h}} = \left(\mathbf{A}_P^H\mathbf{A}_P\right)^{-1}\mathbf{A}_{\hat{\Delta},P}^H\mathbf{r}. \quad (9.13)$$

Note that in case we follow Approach B, the vector $\mathbf{h}$ will be much larger: it will contain $(\Delta_{max} + L_{h_s})N$ entries (rather than $L_{h_s}N$). For large Δ_{max}, Approach B would require significantly more pilot symbols than the approach where we estimate Δ and $\mathbf{h}$ jointly. On the other hand, Approach B would not require the one-dimensional search (9.12) over all possible values of Δ.

9.2.1.2 Method 2: the structured estimator

While (9.12)-(9.13) is conceptually straightforward, there is another way to proceed. Estimating the entire channel vector $\mathbf{h}$ can be avoided by exploiting the underlying channel model (9.3): in principle we only need to estimate $2L$ parameters (the L gains and the L propagation delays), rather than $L_{h_s}N$ channel taps and 1 delay shift. We start from a vector representation of $r_{BB}(t)$, say $\mathbf{r}$. We again apply the ML principle, to obtain a data-aided estimate of the $2L$ channel parameters[3]:

$$\{\hat{\tau}_k, \hat{\alpha}_k\}_{\forall k} = \arg\max_{\{\tau_k,\alpha_k\}_{\forall k}} \log p(\mathbf{r}\,|\mathbf{a}_P, \mathbf{a}_C = \mathbf{0}, \{\tau_k, \alpha_k\}_{\forall k}) \quad (9.14)$$

with

$$\begin{aligned} &\log p(\mathbf{r}\,|\mathbf{a}_P, \mathbf{a}_C = \mathbf{0}, \{\tau_k, \alpha_k\}_{\forall k}) \\ &\propto -\sum_{l_1,l_2=0}^{L-1}\alpha_{l_1}\alpha_{l_2}^*\sum_{n_1,n_2=0}^{P-1} a_{n_1}a_{n_2}^* q\left((n_1-n_2)T + \tau_{l_1} - \tau_{l_2}\right) \\ &+ 2\sum_{l=0}^{L-1}\sum_{n=0}^{P-1}\Re\left\{\alpha_l a_n y^*(nT + \tau_l)\right\} \quad (9.15) \end{aligned}$$

where

$$y(t) = \int_{-\infty}^{+\infty} p^*(-u)\, r_{BB}(t-u)\, du / \sqrt{E_s}$$

[3]The notation $p(\mathbf{r}\,|\mathbf{a}_P, \mathbf{a}_C = \mathbf{0}, \{\tau_k, \alpha_k\}_{\forall k})$ is somewhat misleading, as it gives the impression that the transmitted sequence contains only pilot symbols. The true meaning is as follows: the receiver *operates under the assumption* that the transmitted sequence contains only pilot symbols.

and $q(t) = \int p^*(-u)\,p(t-u)\,du$. Unfortunately (9.15) is difficult to maximize with respect to the gains and delays when $L > 1$. We will introduce a common approximation. When we neglect the cross-terms (i.e., $l_1 \neq l_2$) in the first term in (9.15), we can reduce (9.15) to:

$$\begin{aligned}&\log p\left(\mathbf{r}\,|\mathbf{a}_P, \mathbf{a}_C = \mathbf{0}, \{\tau_k, \alpha_k\}_{\forall k}\right) \\ &\approx \sum_{l=0}^{L-1}\left(-|\alpha_l|^2 \sum_{n=0}^{P-1}|a_n|^2 + 2\Re\left\{\alpha_l \sum_{n=0}^{P-1} a_n y^*(nT+\tau_l)\right\}\right)\end{aligned} \tag{9.16}$$

which is the sum of L terms, each of which depends only on one delay and one complex gain. Hence, the resulting maximization problem can easily be solved. This structured approach is common in DS/SS systems where inter-path-interference is neglected.

Once the channel parameters have been estimated, the overall impulse response $h(t)$ can be reconstructed, so that the signal $r_{BB}(t)$ can further be processed.

9.2.2 Frequency-flat channel

In a frequency-flat channel, we are faced with the following received signal:

$$r_{BB}(t) = Ae^{j\theta}\sum_{n=0}^{N_s-1} a_n p(t-nT-\tau) + w(t) \tag{9.17}$$

where a_n is the n-th data symbol, A is the path amplitude, θ is the carrier phase and τ is the propagation delay. The first P symbols are again pilot symbols. The noise $w(t)$ is complex white Gaussian noise with variance σ^2 per real dimension. We will assume the following a priori distributions:

- A is a real, positive number
- θ is uniformly distributed in $[-\pi, +\pi]$
- τ is uniformly distributed in $[-\Delta_1, +\Delta_2]$, for some known constants Δ_1 and Δ_2.

9.2.2.1 Data-aided estimation

Starting from (9.17), an unstructured channel estimate can be obtained by applying (9.12)-(9.13). Similarly, a structured estimate is obtained from (9.15), for $L = 1$:

$$\begin{aligned}&\log p\left(\mathbf{r}\,|\mathbf{a}_P, \tau, A, \theta\right) \\ &\propto -A^2\sum_{n=0}^{P-1}|a_n|^2 + 2A\sum_{n=0}^{P-1}\Re\left\{a_n e^{j\theta} y^*(nT+\tau)\right\}\end{aligned} \tag{9.18}$$

which is now solved as follows:

$$\hat{\tau} = \arg\max_{\tau\in[-\Delta_1,+\Delta_2]}\left|\sum_{n=0}^{P-1} a_n y^*(nT+\tau)\right| \tag{9.19}$$

$$\hat{\theta} = -\arg\left\{\sum_{n=0}^{P-1} a_n y^*(nT+\hat{\tau})\right\} \tag{9.20}$$

$$\hat{A} = \frac{\Re\left\{\sum_{n=0}^{P-1} a_n e^{j\hat{\theta}} y^*(nT+\hat{\tau})\right\}}{\sum_{n=0}^{P-1}|a_n|^2}. \tag{9.21}$$

9.2.2.2 Other estimation algorithms

By now, the reader might get the impression that all estimation techniques fall into the ML, EM or SAGE category. This is far from true. Contrary to EM or SAGE methods, conventional algorithms are application specific and often of very low complexity. It is simply impossible to give a comprehensive overview of standard estimation methods, due to the huge number of algorithms available in technical literature. The reader is referred to the standard works in estimation and synchronization theory [16,17]. We simply present some simple algorithms relevant to our system model. The entire process is depicted in Fig. 9.3 and consists of both data-aided and non-data-aided algorithms.

Step 0: burst detection

First of all, the receiver measures the energy of the incoming signal. When the receiver expects a burst, a number of samples will be stored for further processing. Since such energy-detection is inherently sub-optimal, the exact time of the beginning of the burst is unknown. The uncertainty is modeled in the distribution of τ. This explains the introduction of the parameters Δ_1 and Δ_2 in the previous section. For convenience, we will assume $\Delta_i/T \in \mathbb{N}$.

Step 1: (Fractional) delay estimation

Secondly, τ is estimated by exploiting the cyclostationarity present in the signal $r_{BB}(t)$ [16]: $r_{BB}(t)$ is filtered with $p^*(-t)$ resulting in a signal $y(t)$. As we will consider only digital implementation, we can start from the samples $y(kT/N)$. We process these samples, yielding an estimate of τ [89]:

$$\hat{\tau} = -\frac{T}{2\pi}\arg\left\{\sum_k F\left(y\left(kT/N\right)\right)\exp\left(-j2\pi\frac{k}{N}\right)\right\} \tag{9.22}$$

where $F(.)$ is a suitable non-linear function. A popular example is $F(x) = |x|^2$, resulting in the well-known Oerder&Meyr synchronizer [90]. Taking into account the Bandpass Sampling Theorem, the Oerder&Meyr synchronizer requires $N \geq 2BT$, where B is the one-sided bandwidth of received signal. It is important to note that the Oerder&Meyr estimator requires no knowledge of A or θ.

The reader may notice that $\hat{\tau} \in [-T/2, +T/2]$. When $\Delta_i > T/2$ $(i = 1, 2)$, which is often the case, $\hat{\tau}$ should be interpreted as an estimate of the fractional part of τ. To clarify this, let us break up τ as follows:

$$\tau = \varepsilon_\tau + k_\tau T \tag{9.23}$$

with $\varepsilon_\tau \in [-T/2, +T/2]$ and k_τ an integer, belonging to the set S_τ, defined as

$$k_\tau \in \left[-\Delta_1/T, \Delta_2/T\right]. \tag{9.24}$$

Hence, estimation of τ can be broken up into estimation ε_τ and estimation of k_τ. The estimate $\hat{\tau}$ from (9.22) should be understood as an estimate of ε_τ, rather than of τ. We will refer to (9.22) as *fractional delay estimation* and write $\hat{\varepsilon}_\tau$ in lieu of the misleading $\hat{\tau}$. The process of determining k_τ is known as *frame synchronization* and is the next step in the estimation process.

Step 2: Frame synchronization

Assuming the estimate of ε_τ to be correct, we now apply (9.19) to find an estimate of k_τ:

$$\hat{k}_\tau = \arg\max_{k_\tau}\left|\sum_{n=0}^{P-1} a_n y^*\left(nT + k_\tau T + \hat{\varepsilon}_\tau\right)\right|. \tag{9.25}$$

This is again a famous algorithm [91], operating by correlating the pilot symbols with the time-shifted symbol-rate matched filter outputs. The final estimate of τ is now given by $\hat{\tau} = \hat{k}_\tau T + \hat{\varepsilon}_\tau$.

Step 3: (Fractional) phase estimation

Once τ has been estimated, matched filter output samples at the symbol rate (i.e., $y(nT + \hat{\tau})$, $n = 0, \ldots, N_s - 1$) are used to estimate θ. Note that when $\hat{\tau} = \tau$, we have $y(nT + \hat{\tau}) = Ae^{j\theta}a_n + n_n$. We will consider the Viterbi&Viterbi phase estimator from [92]:

$$\hat{\theta} = \frac{1}{M_\Omega}\arg\left\{\sum_{k=0}^{N_s}\left(y\left(nT + \hat{\tau}\right)\right)^{M_\Omega}\right\} \tag{9.26}$$

where M_Ω is known as the phase ambiguity number of the constellation Ω, defined as follows: $2\pi/M_\Omega$ is the smallest angle of rotational symmetry[4] of Ω. For M-PSK constellations, $M_\Omega = M$, where M is the number of constellation points. For square QAM constellations, $M_\Omega = 4$.

[4] ψ in an angle of rotational symmetry of Ω when rotating Ω over ψ yields Ω.

Similar to delay estimation, (9.26) provides an estimate in the range $[-\pi/M_\Omega, +\pi/M_\Omega]$. We write θ as

$$\theta = \varepsilon_\theta + 2\pi \frac{k_\theta}{M_\Omega} \tag{9.27}$$

where $\varepsilon_\theta \in [-\pi/M_\Omega, +\pi/M_\Omega]$ and $k_\theta \in \{0, \ldots, M_\Omega - 1\}$. Hence, (9.26) should be interpreted as an estimate of ε_θ, rather than of θ. Estimation of k_θ is referred to as *phase ambiguity resolution*, which is the next step in the estimation process.

Step 4: Phase ambiguity resolution

Similar to frame synchronization, we apply (9.20) to perform phase ambiguity resolution:

$$\hat{k}_\theta \quad = \quad \arg\max_{k_\theta} \Re \left\{ \sum_{n=0}^{P-1} a_n e^{j\hat{\varepsilon}_\theta} e^{j2\pi k_\theta / M_\Omega} y^* \left(nT + \hat{\tau}\right) \right\}. \tag{9.28}$$

The final estimate of θ is now given by $\hat{\theta} = \hat{k}_\theta 2\pi/M_\Omega + \hat{\varepsilon}_\theta$.

Step 5: Amplitude estimation

The last estimation step is estimation of A. A data-aided estimate of A is given by (9.21).

Step 6: Message computation

Once θ, τ and A have been estimated, the samples $\left\{ y\left(nT + \hat{\tau}\right) e^{-j\hat{\theta}} \right\}$ are provided to the factor graph as parameters, and messages $\mu_{a_k \to \varphi}\left(a_k\right)$ can be computed.

9.3 EM-based code-aided estimation

We have seen how a combination of data-aided and non-data-aided techniques can be used to estimate the channel impulse response. However, they are one-shot algorithms: the estimate is provided to the factor graph detector, where iterative equalization, demodulation and decoding is performed. No information flows back from the detector to the estimator.

By applying the EM algorithm, we will introduce code-aided algorithms that are able to exploit information from the detector in an elegant and intuitive way. Before we go into the details of EM-based code-aided estimation algorithms, some important issues need to be addressed:

- What will we estimate? The data symbols or the channel parameters?
- Secondly, what would be an appropriate choice for complete/hidden data?

A most natural way to proceed is to estimate the data sequence while considering the channel parameters as nuisance parameters. Indeed, the final goal of a receiver is recovering the data symbols, not estimating the channel parameters! However, there are several problems with such an approach: in many cases $p\left(\mathbf{d}_c \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$ will be impossible to evaluate, so that the E-step (8.8) can only be performed approximately. Secondly, the M-step: we need to make an estimate (=a hard decision) of the data sequence at each iteration of the EM algorithm. Not only does making such a decision require searching exhaustively the space of all codewords, all state-of-the-art detectors require soft information w.r.t. the coded data symbols to operate properly. Still, many papers on iterative EM-based estimation apply the EM algorithm to estimate the data sequence, rather than the channel parameters [93–95].

Following [19,96–99], we have taken another, at first sight counter-intuitive approach: we treat the data symbols as nuisance parameters in estimating the channel parameters. The complete/hidden data will contain the data sequence and the received signal. It will turn out that the resulting estimation algorithm is able to exploit information from the factor graph detector through the a posteriori probabilities of the coded symbols. This enables the estimator and factor graph detector to cooperate in a seamless way. We follow the same pattern as in the previous section: we first consider a frequency-selective channel and then a frequency-flat channel. We remind again that any technique derived for a frequency-selective channel can be applied without alteration to a frequency-flat channel.

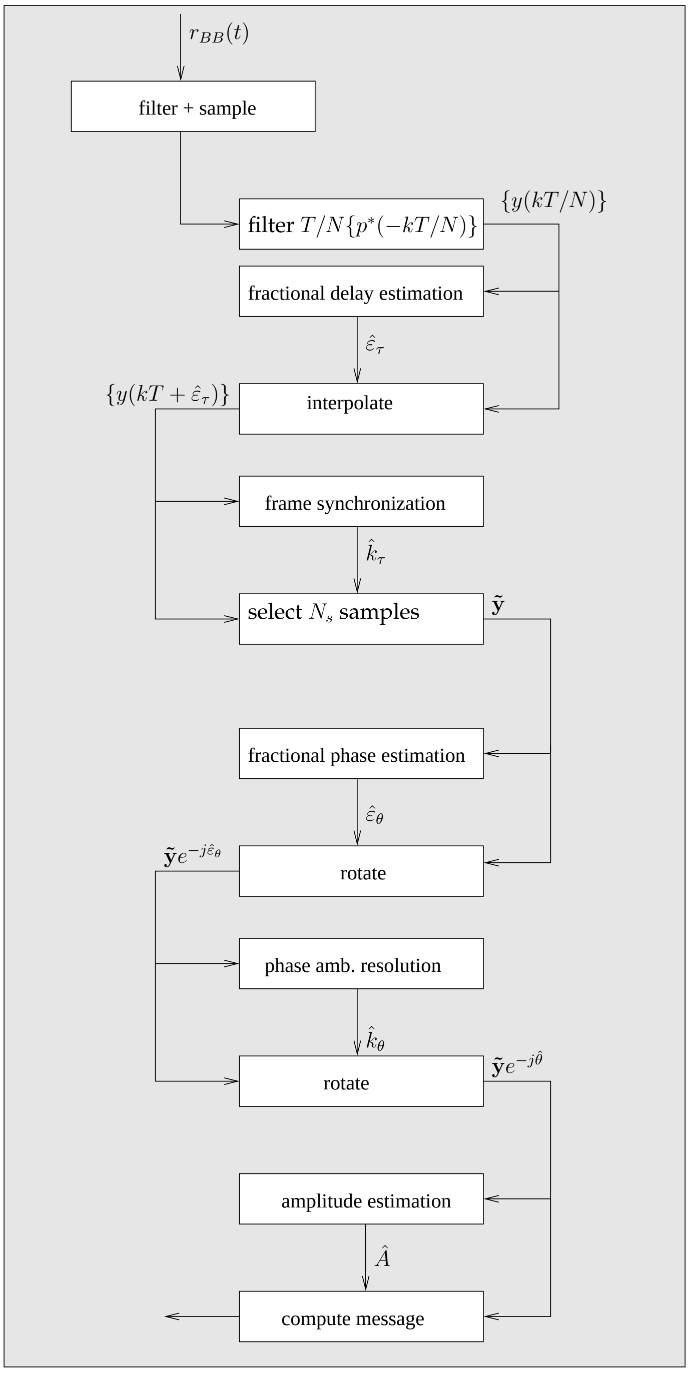


Figure 9.3: *Conventional estimation for flat fading channel, with* $\tilde{\mathbf{y}}_k = y\,(kT + \hat{\tau})$, $k = 0, \ldots, N_s - 1$.

9.3.1 Frequency-selective channel

The signal at the input of the receiver is of the familiar form

$$r_{BB}(t) = \sum_{l=0}^{L-1} \alpha_l \sum_{n=0}^{N_s-1} a_n p(t - nT - \tau_l) + w(t). \tag{9.29}$$

As before, we will describe two estimation techniques. The first one is unstructured and estimates the sampled version of the overall impulse response $h(t)$ directly, similar to the unstructured technique from section 9.2.1.1. The second one exploits the underlying channel model and estimates the channel parameters, similar to section 9.2.1.2.

9.3.1.1 Method 1: the unstructured approach

Starting from (9.6)

$$\begin{aligned} \mathbf{r} &= \begin{bmatrix} \mathbf{0}_\Delta \\ \mathbf{A}_D \\ \mathbf{0}_{(\Delta_{max}-\Delta)} \end{bmatrix} \mathbf{h} + \mathbf{w} & (9.30) \\ &= \mathbf{A}_\Delta \mathbf{h} + \mathbf{w} & (9.31) \end{aligned}$$

and applying the EM algorithm with complete data $\mathbf{d}_c = [\mathbf{a}, \mathbf{r}]$ to estimate $\mathbf{d}_e = [\mathbf{h}, \Delta]$, yields the following E-step (see Eq. (8.13)):

$$\begin{aligned} & Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right) \\ &= E_{\mathbf{a}}\left[\log p(\mathbf{r}\,|\mathbf{a}, \Delta, \mathbf{h}) \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right] & (9.32) \\ &= E_{\mathbf{a}}\left[-\mathbf{h}^H \mathbf{A}_D^H \mathbf{A}_D \mathbf{h} + 2\Re\left\{\mathbf{r}^H \mathbf{A}_\Delta \mathbf{h}\right\} \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right] & (9.33) \end{aligned}$$

E-step We can evaluate $Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right)$ as

$$Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right) = -\mathbf{h}^H \widetilde{\mathbf{A}_D^H \mathbf{A}_D} \mathbf{h} + 2\Re\left\{\mathbf{r}^H \tilde{\mathbf{A}}_\Delta \mathbf{h}\right\} \tag{9.34}$$

where $\tilde{\mathbf{A}}_\Delta = E_{\mathbf{a}}\left[\mathbf{A}_\Delta \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right]$ and $\widetilde{\mathbf{A}_D^H \mathbf{A}_D} = E_{\mathbf{a}}\left[\mathbf{A}_D^H \mathbf{A}_D \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right]$.

M-step Solving (9.34) w.r.t. $\mathbf{h}$ and Δ, yields the following solution of the M-step:

$$\hat{\Delta}^{(i+1)} = \arg\max_{\Delta} \left\{ \Re\left(\mathbf{y}^H \tilde{\mathbf{A}}_\Delta \left(\widetilde{\mathbf{A}_D^H \mathbf{A}_D}\right)^{-1} \tilde{\mathbf{A}}_\Delta^H \mathbf{r} \right) \right\} \tag{9.35}$$

$$\hat{\mathbf{h}}^{(i+1)} = \left(\widetilde{\mathbf{A}_D^H \mathbf{A}_D}\right)^{-1} \tilde{\mathbf{A}}_{\hat{\Delta}^{(i+1)}}^H \mathbf{r} \tag{9.36}$$

The main question is of course, how can the quantities $\tilde{\mathbf{A}}_{\hat{\Delta}^{(i+1)}}^H$ and $\left(\widetilde{\mathbf{A}_D^H \mathbf{A}_D}\right)^{-1}$ be computed? $\tilde{\mathbf{A}}_{\hat{\Delta}^{(i+1)}}$ contains elements of the form

$$\begin{aligned} \tilde{a}_n &\doteq E_{\mathbf{a}}\left[a_n \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right] & (9.37) \\ &= \sum_{\omega \in \Omega} \omega \times p\left(a_n = \omega \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right) & (9.38) \end{aligned}$$

for $n \in \{0, \ldots, N_s - 1\}$. The matrix $\widetilde{\mathbf{A}_D^H \mathbf{A}_D}$ contains elements of the form $E_{\mathbf{a}}\left[a_n a_m^* \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right]$. For $m \neq n$, these elements can be approximated as $E_{\mathbf{a}}\left[a_n a_m^* \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right] \approx \tilde{a}_n \tilde{a}_m^*$, while for $m = n$, we obtain

$$\widetilde{|a_n|^2} = \sum_{\omega \in \Omega} |\omega|^2 \times p\left(a_n = \omega \middle| \mathbf{r}, \hat{\Delta}^{(i)}, \hat{\mathbf{h}}^{(i)}\right). \tag{9.39}$$

This reduces the question to: how can we compute the quantities $\tilde{a}_n$ and $\widetilde{|a_n|^2}$? And what is their interpretation? It is clear that $\tilde{a}_n$ and $\widetilde{|a_n|^2}$ can be computed in terms of the marginal a posteriori probabilities (APPs) of the coded symbols, conditioned on the previous estimate of the $\mathbf{d}_e$. These APPs are of course computed by the factor graph detector structures from Part II of this dissertation. In other words, when we provide the detector with the observation $\mathbf{r}$ and the estimate $\hat{\mathbf{d}}_e^{(i)} = \left[\hat{\mathbf{h}}^{(i)}, \hat{\Delta}^{(i)}\right]$, the detector will compute (iteratively, through the sum-product algorithm) the APPs $p\left(a_n = \omega | \, \mathbf{r}, \hat{\mathbf{h}}^{(i)}, \hat{\Delta}^{(i)}\right)$. Note that for pilot symbols the APPs do not depend on $\hat{\mathbf{d}}_e^{(i)}$: the APPs are either zero or one, and perfectly known to the receiver, so that for pilot symbols $\tilde{a}_n = a_n$ and $\widetilde{|a_n|^2} = |a_n|^2$.

The reader will notice the similarities between the data-aided algorithm (9.12)-(9.13) and the new EM-based algorithm (9.35)-(9.36). An initial estimate for the EM-based algorithm can be obtained from (9.12)-(9.13).

A great deal has been achieved in this short paragraph: an important connection has been made between the EM estimator and the factor graph detectors from Part II. An iterative estimation technique has been derived that exploits information from the pilot symbols and coded symbols in a natural, almost intuitive way: the algorithm is obtained by formally replacing the pilot symbols from a data-aided algorithm with the a posteriori expectation of the coded symbols. These a posteriori expectations go under different names in technical literature: soft symbols, soft symbol decisions, symbol expectations... With all the background knowledge we have amassed regarding ML, EM and SAGE estimation, extending these concepts to structured estimation is a fairly straightforward task.

9.3.1.2 Method 2: structured approach

Using a vector representation of $r_{BB}(t)$, we can transform, with obvious notations, (9.29) into the following vector representation

$$\mathbf{r} = \sum_{l=0}^{L-1} \mathbf{s}_l \left(\mathbf{a}, \alpha_l, \tau_l\right) + \mathbf{w}. \tag{9.40}$$

where $\mathbf{s}_l\left(\mathbf{a}, \alpha_l, \tau_l\right)$ is a vector representation of $\alpha_l \sum_{n=0}^{N_s-1} a_n p\left(t - nT - \tau_l\right)$. Let us contemplate (9.40): we are trying to estimate parameters from the superposition of L signals. Referring to the example from section 8.7 in Chapter 8, we notice that we have in fact already solved this problem by means of the SAGE algorithm! Applying (8.27), with $\mathbf{b}_k = [\alpha_k, \tau_k]$ and $\mathbf{d}_{h,k} = [\mathbf{r}, \mathbf{a}]$, the E-step of the SAGE algorithm is given by

$$Q\left(\mathbf{b}_k | \, \hat{\mathbf{b}}^{(i)}\right) = \tag{9.41}$$

$$E_{\mathbf{a}}\left[-\left\|\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\right\|^2 \Big| \, \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right] + 2\Re\left\{E_{\mathbf{a}}\left[\hat{\mathbf{z}}_k^H\left(\mathbf{a}, \hat{\mathbf{b}}^{(i)}\right)\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right) \Big| \, \mathbf{r}, \hat{\mathbf{b}}^{(i)}\right]\right\} \tag{9.42}$$

where

$$\hat{\mathbf{z}}_k\left(\mathbf{a}, \hat{\mathbf{b}}^{(i)}\right) = \mathbf{r} - \sum_{l \neq k} \mathbf{s}_l\left(\mathbf{a}, \hat{\mathbf{b}}_l^{(i)}\right). \tag{9.43}$$

In our case, since $p(t)$ is a square root Nyquist pulse:

$$\left\|\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right)\right\|^2 = |\alpha_k|^2 \sum_{n=0}^{N_s} |a_n|^2 \tag{9.44}$$

while

$$\hat{\mathbf{z}}_k^H\left(\mathbf{a}, \hat{\mathbf{b}}^{(i)}\right)\mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right) = \tag{9.45}$$

$$\mathbf{r}^H \mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right) - \left(\sum_{l \neq k} \mathbf{s}_l\left(\mathbf{a}, \hat{\mathbf{b}}_l^{(i)}\right)\right)^H \mathbf{s}_k\left(\mathbf{a}, \mathbf{b}_k\right).$$

The last term in (9.45) can be written as[5]

$$\left(\sum_{l\neq k}\mathbf{s}_l\left(\mathbf{a},\hat{\mathbf{b}}_l^{(i)}\right)\right)^H\mathbf{s}_k\left(\mathbf{a},\mathbf{b}_k\right)$$

$$=\alpha_k\sum_{l\neq k}\hat{\alpha}_l^*\sum_{n,n'}a_n^*a_{n'}q\left(nT-n'T+\hat{\tau}_l-\tau_k\right)\tag{9.46}$$

$$=\alpha_k\mathbf{a}^H\mathbf{R}\left(\tau_k,\hat{\mathbf{b}}^{(i)}\right)\mathbf{a}\tag{9.47}$$

where $q\left(t\right)=\int p^*\left(-u\right)p\left(t-u\right)du$ and

$$\mathbf{R}_{n,n'}\left(\tau_k,\hat{\mathbf{b}}^{(i)}\right)=\sum_{l\neq k}\hat{\alpha}_l^*q\left(nT-n'T+\hat{\tau}_l-\tau_k\right).\tag{9.48}$$

Finally, $\mathbf{r}^H\mathbf{s}_k\left(\mathbf{a},\mathbf{b}_k\right)$ is given by

$$\mathbf{r}^H\mathbf{s}_k\left(\mathbf{a},\mathbf{b}_k\right)=\alpha_k\sum_n a_ny^*\left(nT+\tau_k\right)\tag{9.49}$$

where $y\left(t\right)=\int p^*\left(-u\right)r_{BB}\left(t-u\right)du$. After substitution we obtain

$$\begin{aligned}&Q\left(\mathbf{b}_k|\,\hat{\mathbf{b}}^{(i)}\right)\\&=-\left|\alpha_k\right|^2\sum_n E_{\mathbf{a}}\left[\left|a_n\right|^2\middle|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]+2\Re\left\{\alpha_k\sum_n E_{\mathbf{a}}\left[a_n|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]y^*\left(nT+\tau_k\right)\right\}\\&-2\Re\left\{\alpha_kE_{\mathbf{a}}\left[\mathbf{a}^H\mathbf{R}\left(\tau_k,\hat{\mathbf{b}}^{(i)}\right)\mathbf{a}\middle|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]\right\}\end{aligned}\tag{9.50}$$

Note that the first term is equal to $\left|\alpha_k\right|^2N_s$ for M-PSK constellations. $Q\left(\mathbf{b}_k|\,\hat{\mathbf{b}}^{(i)}\right)$ can be computed from the APPs $p\left(a_n|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right)$ when we again use the approximation

$$E_{\mathbf{a}}\left[a_na_m^*|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]\approx E_{\mathbf{a}}\left[a_n|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]E_{\mathbf{a}}\left[a_m^*|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]\tag{9.51}$$

for $n\neq m$. We can now maximize $Q\left(\mathbf{b}_k|\,\hat{\mathbf{b}}^{(i)}\right)$ w.r.t. τ_k and α_k: for a given τ_k a closed form solution for α_k can be found. After back-substitution of this solution into (9.50), we can maximize w.r.t. τ_k with a one-dimensional search.

9.3.2 Frequency-flat channel

By now, it is a straightforward task to derive a similar algorithm for the frequency-flat channel estimator. We start again from the following received signal

$$r_{BB}\left(t\right)=Ae^{j\theta}\sum_{n=0}^{N_s-1}a_np\left(t-nT-\tau\right)+w\left(t\right).\tag{9.52}$$

We simply specialize the multi-path case from section 9.3.1 to $L=1$. In (9.50), the last term can now be dropped since in the definition of $\mathbf{R}\left(\tau_k,\hat{\mathbf{b}}^{(i)}\right)$ from (9.48), there are no terms in the summation. Hence, (9.50) becomes

$$Q\left(\mathbf{b}|\,\hat{\mathbf{b}}^{(i)}\right)=-A^2\sum_n E_{\mathbf{a}}\left[\left|a_n\right|^2\middle|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]+2\Re\left\{Ae^{j\theta}\sum_n E_{\mathbf{a}}\left[a_n|\,\mathbf{r},\hat{\mathbf{b}}^{(i)}\right]y^*\left(nT+\tau\right)\right\}.\tag{9.53}$$

Introducing $\tilde{a}_n=E\left[a_n|\,\mathbf{r},\hat{\mathbf{d}}_e^{(i)}\right]$and $\widetilde{\left|a_n\right|^2}=E\left[\left|a_n\right|^2\middle|\,\mathbf{r},\hat{\mathbf{d}}^{(i)}\right]$, the maximization is now fairly straightforward. The final result is given by (see also [19]):

[5]See Chapter 4, section 4.6 for information regarding vector represenations of signals.

$$\hat{\tau}^{(i+1)} = \arg\max_{\tau} \left| \sum_{n=0}^{N_s-1} \tilde{a}_n y^* \left(nT + \tau\right) \right| \tag{9.54}$$

while

$$\hat{\theta}^{(i+1)} = -\arg\left\{ \sum_{n=0}^{N_s-1} \tilde{a}_n y^* \left(nT + \hat{\tau}^{(i+1)}\right) \right\} \tag{9.55}$$

and

$$\hat{A}^{(i+1)} = \frac{\sum_{n=0}^{N_s-1} \Re\left\{ e^{j\hat{\theta}^{(i+1)}} \tilde{a}_n y^* \left(nT + \hat{\tau}^{(i+1)}\right) \right\}}{\sum_{n=0}^{N_s-1} \widetilde{|a_n|^2}}. \tag{9.56}$$

The reader will notice (again) the striking similarities between (9.19) and (9.54), between (9.20) and (9.55) and between (9.21) and (9.56). Indeed, the code-aided estimation algorithms are again obtained by formally replacing pilot symbols with the corresponding a posteriori expectations.

Initialization

The EM algorithm requires an initial estimate of $\mathbf{d}_e$. This estimate can be obtained through a conventional estimation algorithm, as described in section 9.2.2.

9.3.3 Two extreme cases

9.3.3.1 Data-aided estimation

EM and SAGE estimators can be constructed to exploit solely pilot symbols. In this case, we simply ignore the unknown data symbols (replace them with zeros in the receiver's observation model) and place the pilot symbols in the complete/hidden data space. This technique could be used, for instance, to refine a data-aided estimate of the channel parameters for a frequency-selective channel, by incorporating the effect in inter-path-interference. In fact, most existing algorithms for multi-path channel estimation that use SAGE are data-aided [100, 101].

9.3.3.2 Uncoded transmission

In some cases, decoding the packet is a very expensive operation and one may want to treat the data as uncoded during the estimation stage. This can be achieved by removing, from the factor graph of the receiver (Fig. 9.1), that part of the graph that corresponds to the error-correcting code $I\left[\mathbf{c} = \chi\left(\mathbf{b}\right)\right]$ and the part corresponding to the a priori distribution $p\left(\mathbf{b}\right)$.

9.4 Sum-product code-aided estimation

As we have mentioned in Chapter 8, section 8.4, we can create a factor graph that includes the unknown channel parameters and transmit mode parameters as variables (i.e., as edges). Although the exact execution of the sum-product algorithm is generally fairly complex, the idea of performing joint estimation and detection through factor graphs has received increased attention from the technical community. Interesting papers in this area can be found in [102–104] (for phase estimation) and [105] (for auto-regressive model parameter estimation). These works capitalize on an earlier paper [106], where a general framework for iterative receiver design using factor graphs was described and applied to channel estimation. Also in [106] the problem related to message representation is tackled: the authors propose to use canonical distributions. This idea was adopted in [102, 104, 105], where messages are represented by distributions that can be described with few parameters (e.g., a mixture of Gaussian distributions) or by resorting to Monte Carlo methods.

A more pragmatic approach was taken in [30, 107] and [85]: the authors break up the overall factor graph into smaller graphs. Within each graph the sum-product algorithm is applied. However, messages between graphs are of a different nature (e.g., hard or soft decisions).

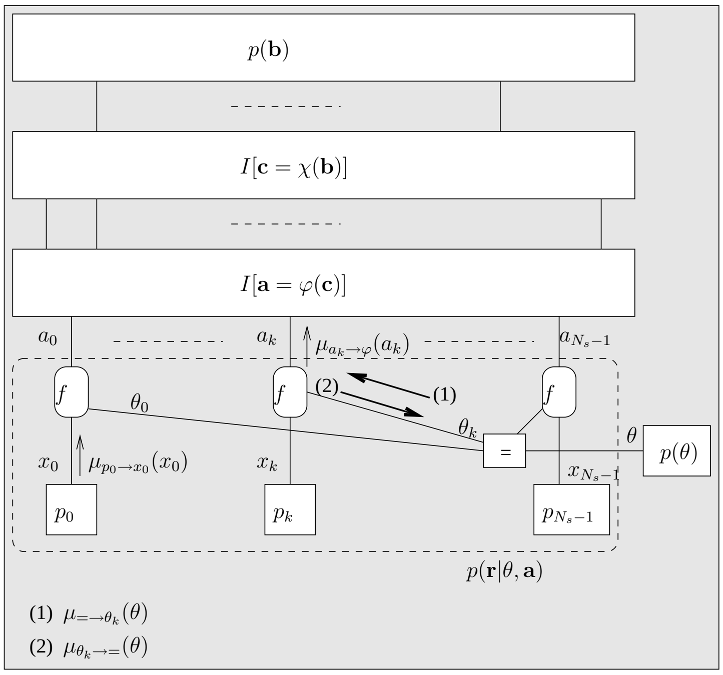

$p(\mathbf{b})$
$I[\mathbf{c}=\chi(\mathbf{b})]$
$I[\mathbf{a}=\varphi(\mathbf{c})]$
a_0
a_k
$\mu_{a_k\to\varphi}(a_k)$
a_{N_s-1}
f
θ_0
(2)
(1)
θ_k
$=$
θ
$p(\theta)$
x_0
$\mu_{p_0\to x_0}(x_0)$
x_k
x_{N_s-1}
p_0
p_k
p_{N_s-1}
$p(\mathbf{r}|\theta,\mathbf{a})$
(1) $\mu_{=\to\theta_k}(\theta)$
(2) $\mu_{\theta_k\to=}(\theta)$

Figure 9.4: *Factor graph representation of* $p(\mathbf{b},\mathbf{c},\mathbf{a},\mathbf{x},\theta\,|\mathbf{r})$ *with unknown carrier phase* θ

9.4.1 Example

A most insightful example is that of phase estimation. Assume we are given an observation

$$\mathbf{r}=\mathbf{a}e^{j\theta}+\mathbf{w} \tag{9.57}$$

where $\mathbf{a}$ is the sequence of N_s coded data symbols, θ is the carrier phase and $\mathbf{w}$ is a vector of iid AWGN samples with $E\left[\mathbf{w}\mathbf{w}^H\right]=2\sigma^2\mathbf{I}_{N_s}$. Introducing $\mathbf{x}=\mathbf{a}e^{j\theta}$, we can construct a factor graph of

$$p\left(\mathbf{b},\mathbf{c},\mathbf{a},\mathbf{x},\theta\,|\mathbf{r}\right)=\prod_{k=0}^{N_s-1}p\left(r_k\,|x_k\right)I\left[x_k=a_ke^{j\theta}\right]I\left[\mathbf{a}=\varphi\left(\mathbf{c}\right)\right]I\left[\mathbf{c}=\chi\left(\mathbf{b}\right)\right]p\left(\mathbf{b}\right)p\left(\theta\right) \tag{9.58}$$

where φ is the function that maps bits to constellation points, χ represents the transformation from information bits to coded bits and

$$p\left(r_k\,|x_k\right)\propto\exp\left(-\frac{1}{2\sigma^2}\left|r_k-x_k\right|^2\right). \tag{9.59}$$

The corresponding factor graph is shown in Fig. 9.4, where we have abbreviated $p\left(r_k\,|x_k\right)$ by p_k. The nodes marked f, represent the function

$$f\left(a_k,x_k,\theta_k\right)=I\left[x_k=a_ke^{j\theta_k}\right] \tag{9.60}$$

while the equality node assures all θ_k are equal to θ.

Applying the sum-product algorithm yields

1. A message is sent from the all the nodes of degree one:

$$\begin{aligned}\mu_{p(\theta)\to\theta}\left(\theta\right)&=p\left(\theta\right)\\ \mu_{p_k\to x_k}\left(x_k\right)&=p\left(r_k\,|x_k\right),\forall k\end{aligned}$$

2. The equality-node sends messages over all the θ_k-edges[6]:

$$\begin{aligned}
\mu_{\boxed{=}\to\theta_k}(\theta_k) &= \int_0^{2\pi} I\left[\theta=\theta_k=\theta_{l\neq k}\right]\prod_{l\neq k}\mu_{f\to\theta_l}(\theta_l)\,\mu_{p(\theta)\to\theta}(\theta)\,d\theta d\theta_{l\neq k}\\
&= \int_0^{2\pi}\delta(\theta-\theta_k)\,\mu_{p(\theta)\to\theta}(\theta)\prod_{l\neq k}\mu_{f\to\theta_l}(\theta_l)\,\delta(\theta_l-\theta_k)\,d\theta d\theta_{l\neq k}\\
&= \prod_{l\neq k}\mu_{f\to\theta_l}(\theta_k)\,\mu_{p(\theta)\to\theta}(\theta_k)
\end{aligned}$$

When $\mu_{f\to\theta_l}(\theta_k)$ is not available, $\mu_{\boxed{=}\to\theta_k}(\theta_k)=\mu_{p(\theta)\to\theta}(\theta_k)$.

3. Messages over the a_k-edges are sent

$$\begin{aligned}
\mu_{a_k\to\varphi}(a_k) &= \int I\left[x_k=a_k e^{j\theta_k}\right]\mu_{p_k\to x_k}(x_k)\,\mu_{\boxed{=}\to\theta_k}(\theta_k)\,dx_k d\theta_k\\
&= \int \delta\left(x_k-a_k e^{j\theta_k}\right)\mu_{p_k\to x_k}(x_k)\,\mu_{\boxed{=}\to\theta_k}(\theta_k)\,dx_k d\theta_k\\
&= \int_0^{2\pi}\mu_{p_k\to x_k}\left(a_k e^{j\theta_k}\right)\mu_{\boxed{=}\to\theta_k}(\theta_k)\,d\theta_k
\end{aligned}$$

4. Now messages are propagated upward from the node marked $I\left[\mathbf{a}=\varphi(\mathbf{c})\right]$ and further on in the graph.

5. Later in the sum-product algorithm, downward messages $\mu_{\varphi\to a_k}(a_k)=\mu_{a_k\to f}(a_k)$ over the a_k-edges will be sent. Now messages can be sent to the phase-equality-node

$$\mu_{\theta_k\to\boxed{=}}(\theta_k)=\sum_{a_k}\mu_{a_k\to f}(a_k)\,\mu_{p_k\to x_k}\left(a_k e^{j\theta_k}\right)$$

6. In the equality node, these messages are multiplied to update $\mu_{\boxed{=}\to\theta_k}(\theta_k)$. Then we go back to step 2.

7. And so forth

Immediately, a problem arises: since θ is defined over a continuous domain, summations have become integrals. How are these integrals computed? Connected to this is the question of how messages should be represented. Keeping track of arbitrary messages is impossible. It is clear that even for this very simple estimation problem, the sum-product algorithm runs into a number of difficulties, due to the domain over which the parameter is defined. This makes joint decoding and estimation through factor graphs unattractive from a complexity point of view. A common way (see also Chapter 8, section 8.4) to avoid the evaluation of the exact messages is to replace $\mu_{\boxed{=}\to\theta_k}(\theta_k)$ with a Dirac distributions $\delta(\theta_k-\theta^*)$, where θ^* is a hard estimate of θ, for instance obtained by maximizing the real message $\mu_{\boxed{=}\to\theta}(\theta)$. It is easily verified that the use of Dirac distribution results in an SP algorithm that requires no integrations.

Note that sum-product code-aided estimation on factor graphs is not really an estimation technique: the goal is the computation of the a posteriori probabilities of the information bits. The messages related to the phase are simply by-products of the sum-product algorithm. This technique will not be pursued further in this dissertation, except in the context of discrete parameter estimation (where integrals are again replaced by summations), in Chapter 11.

9.5 Main points

In this chapter, we have derived estimation algorithms for frequency-selective and frequency-flat channels. We started with conventional data-aided algorithms for a frequency-selective channel: an unstructured approach results in an estimate of a sampled version of the overall impulse response $h(t)$. A structured approach can be used to find estimates of the underlying channel parameters (i.e., the L propagation delays and the L complex gains). We have specialized these algorithms to a frequency-flat channel and included some well-known non-data-aided algorithms for fractional delay and fractional phase estimation.

We have then described a class of estimation algorithms based on the iterative EM and SAGE algorithms for both frequency-selective and frequency-flat channels. It turns out that these algorithms require a posteriori probabilities (APPs) of the coded symbols, conditioned on the previous estimate of the channel parameters. These APPs can be

[6]Note that the indicator function $I\left[\theta=\theta_k=\theta_{l\neq k}\right]$ is now actually a Dirac distribution: $\delta(\theta-\theta_k)\prod_{l\neq k}\delta(\theta_l-\theta_k)$.

computed by the factor graph of the receiver. In many cases the resulting algorithms can be obtained from the corresponding data-aided algorithms, by simply replacing the unknown symbols with so-called a posteriori expectations.

Finally, we mentioned the possibility of performing joint estimation and decoding by incorporating the unknown channel parameters into the factor graph of the receiver. Although this solution is most elegant, the computational complexity is extremely high. As an additional problem, such a factor graph solution requires severe modifications to the detector. This is in stark contrast to the EM-based estimator, which can be combined with an off-the-shelf detector.

∽

In the next chapter we will make some subtle alterations to the EM-based estimation algorithms, resulting in two novel versions of the EM algorithm. In the subsequent chapter these novel algorithms are applied to a set of problems that the EM algorithm traditionally avoids: the estimation of discrete parameters. This includes the abovementioned problems of frame synchronization and phase ambiguity resolution, but also the detection of the transmit mode.

To avoid confusion, we will name the factor graph representation of $p\left(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x} \left| \mathbf{r} \right.\right)$ where $\mathbf{x}$ contains additional parameters not present in $\mathbf{d}_e$ ($\mathbf{d}_e$ is the set of channel parameters, synchronization parameters and transmit mode parameters) the *synchronized factor graph* (such as Fig. 9.1). On the other hand, the factor graph representation of $p\left(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{d}_e \left| \mathbf{r} \right.\right)$ will be named the *overall factor graph* (such as Fig. 9.4).

Chapter 10

Code-aided estimation: the smaller picture

10.1 Introduction

In the previous chapter we have described how the EM algorithm may be applied to perform code-aided estimation in a systematic way. Although the proposed algorithms seem to have solved all our problems, we are not out of the woods just yet. First of all, we have not considered the problem of transmit mode detection. Secondly, we will show that the EM and SAGE code-aided estimation algorithms are still too complex for a practical system. Thirdly, the issue of convergence needs to be addressed. All these and more will be covered in the current chapter. We start with the issue of computational complexity, followed by a discussion of convergence problems. It turns out that convergence is closely related to the problem of estimating discrete parameters.

In short, this chapter will describe some important modifications to the algorithms from the previous chapter, in order to make them more suitable for a practical system.

10.2 Computational complexity

10.2.1 The EM algorithm

In the EM algorithm, each time we update the estimate of $\mathbf{d}_e$, we need to re-compute the APPs $p\left(a_k \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$. To compute these APPs, two steps are required: first of all, the synchronized factor graph[1] of the receiver needs to be reset (see Chapter 3, section 3.2.3). This means that all messages corresponding to cycles are reset to constant messages (uniform pmfs). Secondly, based on the observation $\mathbf{r}$ and the current estimate of $\mathbf{d}_e$ (i.e., $\hat{\mathbf{d}}_e^{(i)}$), the sum-product algorithm is applied to the synchronized factor graph of the receiver. After many iterations (say N_D) within this synchronized factor graph, reliable approximations of the APPs are delivered to the EM estimator and $\hat{\mathbf{d}}_e^{(i+1)}$ is computed. This situation is shown in Algorithm 3. Note that at the end of each decoding stage we check whether or not the decoded word $\hat{\mathbf{a}}$ corresponds to a valid codeword[2]. Since mapped codewords are sparse in Ω^{N_s}, the probability of moving from one codeword to another during detection is negligible. Such a stopping criterion is commonly applied to LDPC codes, but it is argumented in [108] that this technique should be applied to all codes. The reason lies in the fact that (1) checking if a word is a codeword is generally an inexpensive operation, (2) it may reduce decoding time significantly, especially for low BER scenarios.

Now, suppose the receiver has a reliable estimate of $\mathbf{d}_e$ at its disposal and that it requires N_D iterations to compute reliable approximations of the APPs. If each iteration takes T_1 seconds, it takes a total of $N_D T_1$ seconds to detect the packet when we perform one-shot estimation of $\mathbf{d}_e$. On the other hand, if we perform N_{EM} EM iterations to obtain more and more reliable estimates of $\mathbf{d}_e$, the total processing time grows to $N_D N_{EM} T_1$ seconds. This is clearly unacceptable.

Solution - embedded estimation

The most elegant way around this problem is by introducing the concept of *embedded* estimation [109]. The resulting algorithm is shown in Fig. 4. The changes are quite subtle: we reset the synchronized factor graph only once, prior

[1]As defined at the end of the previous chapter: a factor graph where channel parameters and synchronization parameters do *not* appear as variables (i.e., edges).

[2]$\hat{\mathbf{a}}$ is valid codeword $\iff \hat{\mathbf{a}} = \varphi\left(\chi\left(\mathbf{b}\right)\right)$, for some $\mathbf{b}$, with $\hat{\mathbf{a}} = \left[\hat{a}_0, \ldots, \hat{a}_{N_s-1}\right]^T$, where $\hat{a}_k = \arg\max_{a \in \Omega} p\left(a_k = a \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$.

Algorithm 3 EM estimation

```
1: input: observation $\mathbf{r}$, initial estimate $\hat{\mathbf{d}}_e^{(0)}$
2: for $i$ in $0$ to $N_{EM} - 1$ do
3:     reset synchronized factor graph of detector
4:     perform iterative detection with $N_D$ iterations to compute APPs $p\left(a_k \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$
5:     if $\hat{\mathbf{a}}$ is valid codeword then
6:         break
7:     end if
8:     compute $Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right)$
9:     find maximum: determine $\hat{\mathbf{d}}_e^{(i+1)}$
10: end for
```

Algorithm 4 EM embedded estimation

```
1: input: observation $\mathbf{r}$, initial estimate $\hat{\mathbf{d}}_e^{(0)}$
2: reset synchronized factor graph of detector
3: for $i$ in $0$ to $N_{EM} - 1$ do
4:     perform iterative detection with $K_D$ iterations to compute APPs $p\left(a_k \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$
5:     if $\hat{\mathbf{a}}$ is valid codeword then
6:         break
7:     end if
8:     compute $Q\left(\mathbf{d}_e \middle| \hat{\mathbf{d}}_e^{(i)}\right)$
9:     find maximum: determine $\hat{\mathbf{d}}_e^{(i+1)}$
10: end for
```

to the iterative process. Also, when computing the APPs, we perform only a limited number of iterations (say $K_D < N_D$) within the synchronized factor graph. This means we no longer compute the APPs $p\left(a_k \middle| \mathbf{r}, \hat{\mathbf{d}}_e^{(i)}\right)$, but rather a distribution depending on all previous estimates. Since the APPs we computed were only approximations of the true APPs to begin with, these modifications may have little impact on the overall performance. Clearly, embedded estimation is especially attractive when the sum-product algorithm on the synchronized factor graph is itself iterative. When the synchronized factor graph does not contain any cycles, embedded estimation reduces to the conventional EM algorithm, so no gain in terms of computational complexity can be obtained.

Returning to our example, we set $K_D = 1$. Performing N_{EM} EM iterations requires a total of N_{EM} iterations within the synchronized factor graph. So, after $N_{EM}T_1$ seconds, the packet is decoded and, at the same time, the parameter $\mathbf{d}_e$ has been updated N_{EM} times.

Unfortunately, embedded estimation comes with a hidden cost: for some estimation problems, embedded estimation leads to biased[3] estimates, which in turn impact the BER performance [99].

10.2.2 The SAGE algorithm

Of course, the SAGE algorithm suffers from exactly the same problem as the EM algorithm. However, even with embedded estimation, SAGE estimation has another drawback. Each time the APPs are computed (or *updated* would be a more apt term in the case of embedded estimation), we can update only a single component $\mathbf{d}_{k,e}$ of $\mathbf{d}_e$.

As before, we will denote by N_D the number iterations required to compute reliable APPs in the synchronized factor graph and by T_1 the processing time related to a single one of these N_D iterations. Suppose there are $\mathbf{d}_e$ is broken into L parameter-components. We wish to update each of these parameter components N_{EM} times. Then the total processing time becomes a massive $LN_{EM}N_DT_1$ seconds for the SAGE algorithm. With embedded estimation, the processing time would drop to $LN_{EM}T_1$ seconds. Still, this is too large for a practical system.

Solution - multiple parameter updates

A pragmatic solution would be to update multiple parameters, say K_P, based on the same APPs. In our example, if we update all L parameters once for fixed APPs (i.e., $K_P = L$), the total processing time would again be equal to $N_{EM}T_1$ seconds.

[3]An estimate is biased when the estimation error has a non-zero mean.

10.2.3 Further approximations

Many variations can be conceived to reduce the computational complexity. For instance, one could perform EM estimation without exploiting the code to improve the initial estimate. After this, the code-aided EM-estimation algorithm kicks in. One could switch between exploiting code properties for some iterations and not exploiting them for other iterations, etc. Each of these implementations will trade performance for computational complexity.

10.3 Convergence properties

In Chapter 8 we have seen that the EM algorithm delivers estimates with an ever increasing likelihood (or a posteriori probability for MAP estimation). In the same chapter, it was shown that the ML or MAP estimate is always a solution of the EM algorithm. These two statements do *not* imply that the EM algorithm always converges to the ML of MAP estimate. Many technical papers that deal with code-aided estimation disregard this fact and merely state that 'EM converges to ML under some mild conditions'. These mild conditions are related to the initial estimate. Because the impact of the initial estimate has been mostly disregarded in technical literature, several types of estimation problems could not be solved by the EM approach. In this section, we delve deeper into the problems related to the initial estimate.

10.3.1 Convergence characteristic

Our goal is to (a) graphically describe the evolution of the parameter estimate from one iteration to the next and (b) to investigate properties of solutions of the EM algorithm. We constrain ourselves to the estimation of a scalar parameter.

We estimate a parameter d_e through an iterative estimation algorithm according to the update rule $\hat{d}_e^{(i+1)} = f\left(\hat{d}_e^{(i)}\right)$. We start from an initial estimate $\hat{d}_e^{(0)}$, and compute a sequence of estimates $\hat{d}_e^{(1)}, \hat{d}_e^{(2)}, \ldots$ We then define the estimation error at the i-th iteration as $e^{(i)} = \hat{d}_e^{(i)} - d_e$. Hopefully $e^{(i)}$ will become very small as i increases. We also know that any solution of an iterative estimation algorithm satisfies $\hat{d}_e = f\left(\hat{d}_e\right)$, i.e., it is a fixed point of the update rule. Let us try to visually represent this information. We first define two additional parameters $\eta(e, d_e)$ and $\xi(e, d_e)$, as described in Algorithm 5.

Algorithm 5 Convergence characteristic

```
 1: for $e$ in $-\infty$ to $+\infty$ do
 2:     set $\hat{d}_e^{(i)} = e + d_e$
 3:     reset the synchronized factor graph of the receiver
 4:     generate a suitable observation $\mathbf{r}$, given $d_e$
 5:     perform the sum-product algorithm
 6:     compute $\hat{d}_e^{(i+1)} = \arg\max_{d_e} Q\left(d_e \middle| \hat{d}_e^{(i)}\right)$
 7:     compute $\eta(e, d_e) \doteq e^{(i+1)} = \hat{d}_e^{(i+1)} - d_e$
 8:     compute $\xi(e, d_e) \doteq Q\left(\hat{d}_e^{(i)} \middle| \hat{d}_e^{(i)}\right) = Q(e + d_e | e + d_e)$
 9:     store $\eta(e, d_e)$ and $\xi(e, d_e)$
10: end for
```

- $\eta(e, d_e)$ should be interpreted as the estimation error at the *current* iteration, given that the *previous* estimate of d_e had estimation error e.

- $\xi(e, d_e)$ is simply the value of the Q-function, evaluated in the previous estimate.

Note that, by construction, $\xi(e, d_e)$ and $\eta(e, d_e)$ do not depend on the iteration index i. We then plot, for a fixed d_e, $E[\eta(e, d_e)] - e$ and $E[\xi(e, d_e)]$ as a function of e. Here, the expectations are taken over the observations $\mathbf{r}$. Then, from $E[\eta(e, d_e)] - e$ we can investigate the evolution of the iterative estimation algorithm, while $E[\xi(e, d_e)]$ allows us to interpret properties of the solutions of the estimation algorithm.

The definitions of $\eta(e, d_e)$ and $\xi(e, d_e)$ may be somewhat contrived at first sight. We will now consider an example to illuminate their use.

10.3.2 Example: phase estimation

To focus our attention, let us consider an example: phase estimation $d_e = \theta$. In this case $\eta(e, d_e)$ and $\xi(e, d_e)$ do not depend on d_e. Let us also consider a specific turbo code with QPSK mapping[4].

If Fig. 10.1, we plot $E[\eta(e, 0)] - e$ as a function of e for different SNRs. How to interpret this plot? We observe a number of zero-crossings, namely at $e \in \{-\pi, -3\pi/4, -\pi/2, -\pi/4, 0, \pi/4, \pi/2, 3\pi/4, \pi\}$, independent of the SNR. For these zero-crossings, we have $E[\eta(e, d_e)] - e = 0$, or $E[\eta(e, d_e)] = e$. Hence, these points correspond to fixed points (solutions) of the EM algorithm: when e is such a fixed point, the estimate of d_e, on average, remains fixed. The set of fixed points can be broken up into two subsets: those corresponding to positive zero-crossings (i.e., $e \in \{-3\pi/4, -\pi/4, \pi/4, 3\pi/4\}$) and those corresponding to negative zero-crossings (i.e., $e \in \{-\pi, -\pi/2, 0, \pi/2, \pi\}$). The positive zero-crossings correspond to unstable fixed points: suppose e_0 is such a point. Then, for some small δ, $e = e_0 + \delta$ leads to $E[\eta(e, d_e)] > e$, for $\delta > 0$ and $E[\eta(e, d_e)] < e$ for $\delta < 0$. Hence, the estimation errors move *away from* e_0. On the other hand, negative zero-crossings correspond to stable fixed points: suppose e_0 is such a point. Then, for some small δ, $e = e_0 + \delta$ leads to $E[\eta(e, d_e)] < e$, for $\delta > 0$ and $E[\eta(e, d_e)] > e$ for $\delta < 0$. Hence, the estimation errors move *towards* e_0. Only for estimates within the range $[-\pi/4, +\pi/4]$ will the estimation errors converge to the desired (stable) fixed point: at $e = 0$. For estimates outside this range, it is unlikely that the solution of the EM algorithm will give rise to a small estimation error. This observation is of course also true for the initial estimate. Hence, the range $[-\pi/4, +\pi/4]$ corresponds to the *acquisition range* of the EM algorithm. This means that the EM algorithm must be provided an initial estimate within $[-\pi/4, +\pi/4]$ if we want to have a good chance that the solution of the EM algorithm coincides with the ML/MAP estimate. Additionally, we can infer from Fig. 10.1 that more reliable estimates lead to faster convergence, especially at increased SNR.

In Fig. 10.2, we show $E[\xi(e, d_e)]$ as a function of e. We see that the abovementioned stable (unstable) points coincide with local maxima (minima) of $E[\xi(e, d_e)]$. The global maximum coincides with $e = 0$. For increased SNR, the locations of extrema do not change, but the global maximum becomes more pronounced. Note that the solution of the EM algorithm giving rise to the *global* maximum of $E[\xi(e, d_e)]$ has the smallest estimation error.

Finally, as an illustration, we show in Fig. 10.3 some results for a short random code over QPSK (20 random words in Ω^{10}), for which ML evaluation is possible. On the left side, we show the likelihood function $p(\mathbf{r}|\,\theta)$, while the right side corresponds to $Q(\theta|\,\theta)$. The true phase was $\theta = 0$. Observe the striking similarities between the likelihood function and the Q-function. Note especially the locations of the extrema: they are the same for both figures.

10.3.3 Conclusion

Although the above exposition is somewhat anecdotal, several things have become clear: first of all, the EM algorithm is very sensitive to the initial estimate. A more reliable initial estimate leads to faster convergence, while a less reliable estimate will result in slow convergence. When the initial estimate lies outside the acquisition region of the EM algorithm, the estimates will generally converge to an incorrect solution. For phase estimation this means that the EM algorithm cannot be used to perform joint phase estimation and phase ambiguity resolution, without resorting to pilot symbols. We could have presented similar results for delay estimation; the conclusions would be the same (our results are published in [110]): the EM algorithm cannot be used to perform joint delay estimation and frame synchronization, without resorting to pilot symbols.

Now the good news: stable fixed points correspond to local maxima of $Q(d_e\,|d_e)$, while unstable fixed points correspond to local minima of $Q(d_e\,|d_e)$. Most strikingly, the value of d_e that gives rise to the global maximum of $Q(d_e\,|d_e)$, seems to be the solution with the smallest estimation error. The question arises: how can we exploit this new knowledge?

10.4 Extensions of the EM algorithm

Now that we have some more insight in the convergence behavior of EM estimation algorithms, we define some variations of the EM algorithm that allow us to circumvent the convergence problems, albeit at the cost of some increased computational complexity.

10.4.1 The parallel EM algorithm

The parallel EM algorithm, or P-EM algorithm, refers to the parallel execution of multiple (say M_{EM}) instantiations of the EM algorithm, each with a different initial estimate. These initial estimates give rise to M_{EM} solutions[5] of

[4]Parameters: PCCC, RSC constituent convolutional codes, rate 1/2, generator polynomials $(21, 37)_8$, pseudo-random interleaver of size 334 bits, overall rate 1/3, Gray mapping. Only the first convolutional code is terminated.

[5]Some of these solutions may coincide.

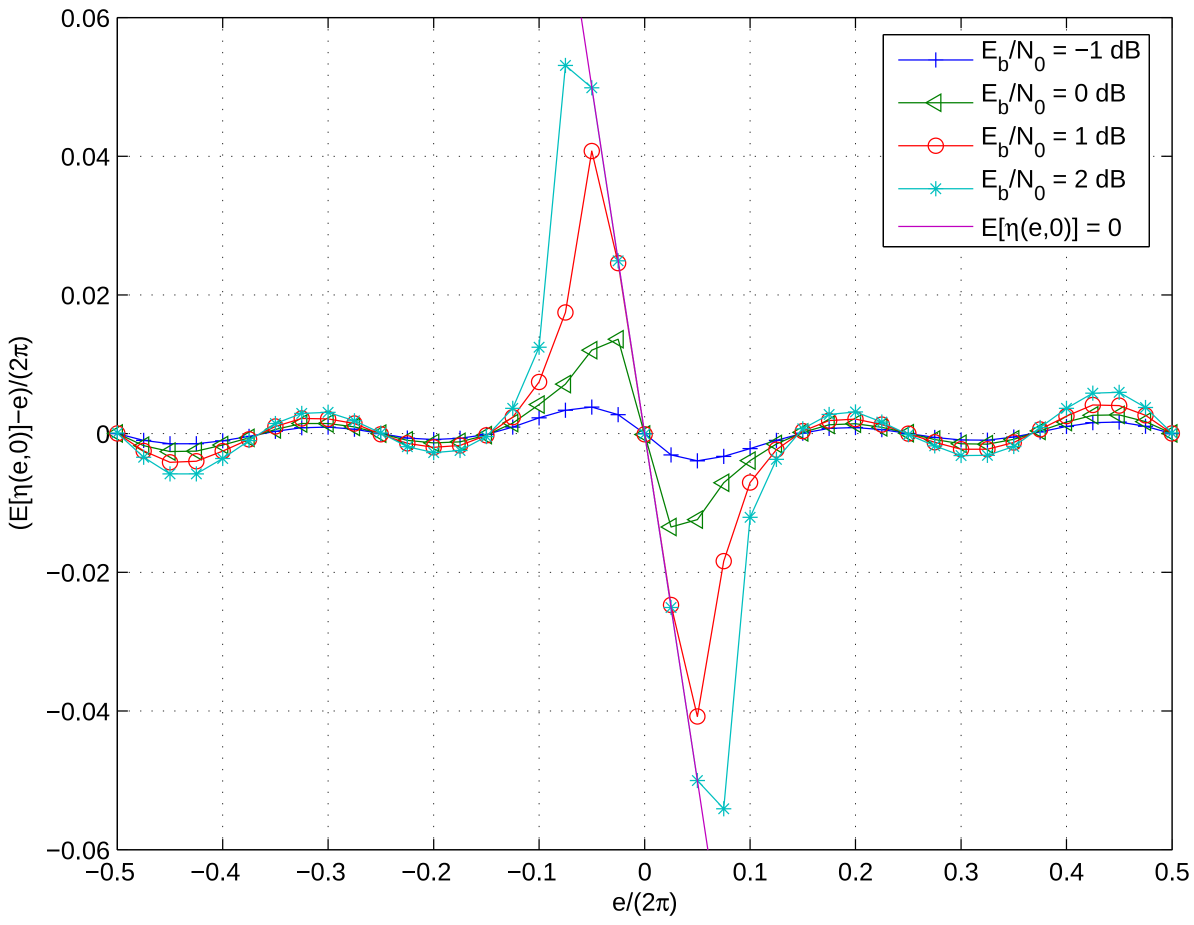


Figure 10.1: *Phase estimation: convergence behavior*

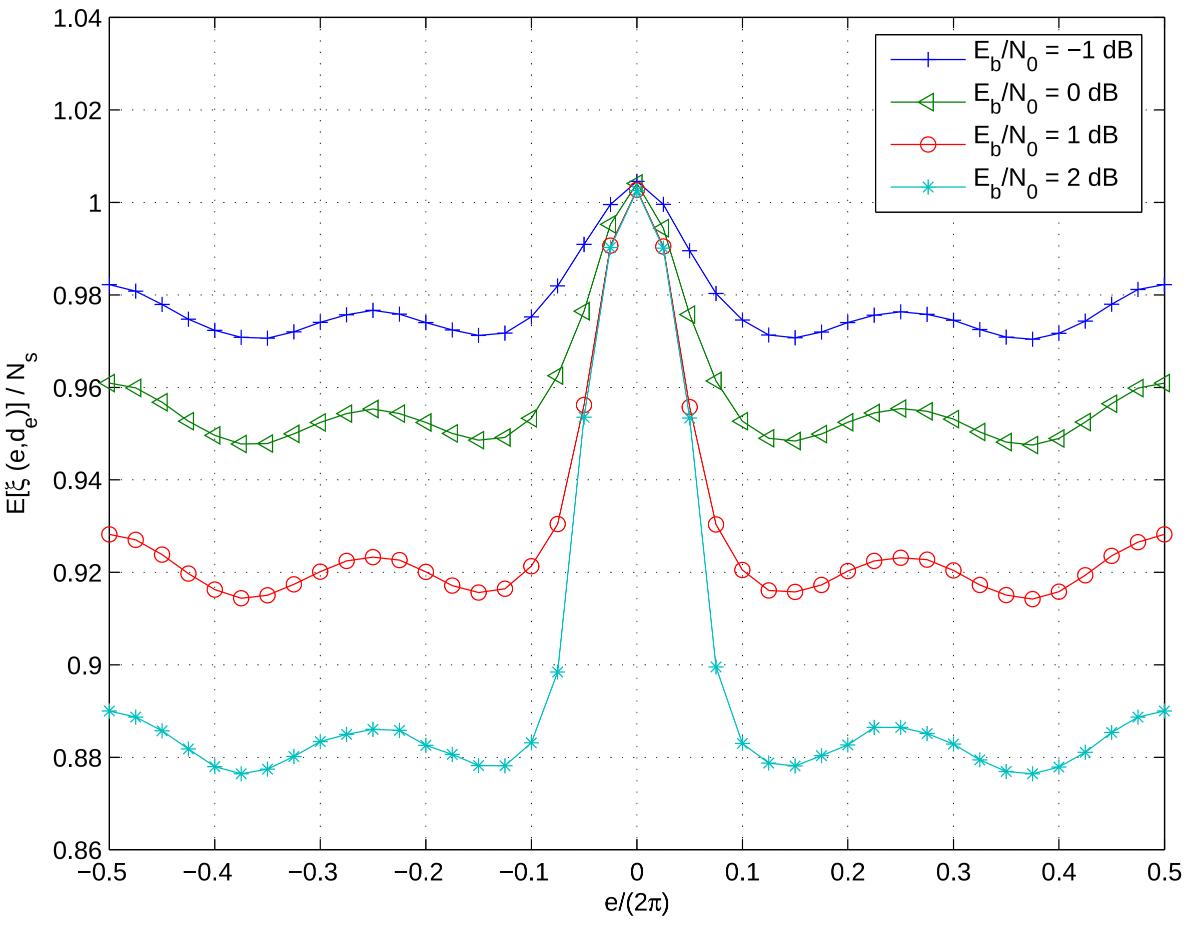


Figure 10.2: *Phase estimation: solutions of the EM algorithm.*

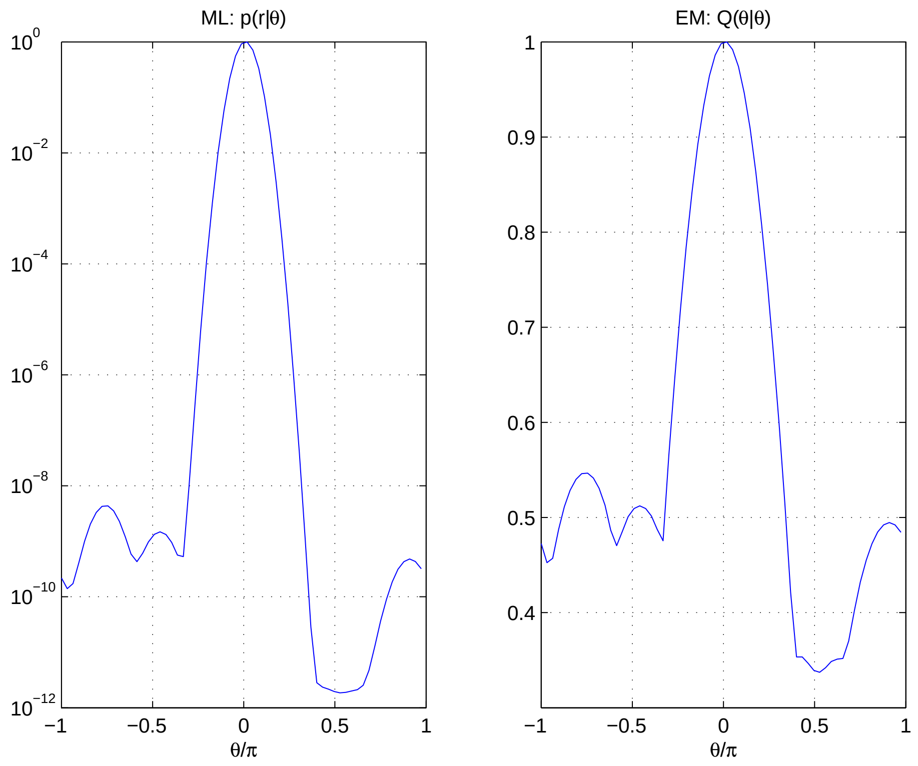


Figure 10.3: *Phase estimation for a short random code of length $N_s = 10$ with 20 codewords: ML vs. EM estimation*

the EM algorithm, $\left[\hat{\mathbf{d}}_{e,0}^{(+\infty)}, \ldots, \hat{\mathbf{d}}_{e,M_{EM}-1}^{(+\infty)}\right]$. We then try to decide which is the 'best' solution of the EM algorithms. According to the MAP (ML) criterion, all we have to do is select the estimate with the largest a posteriori probability (likelihood for ML). In the case of MAP, this means

$$\hat{\mathbf{d}}_e = \arg \max_{\hat{\mathbf{d}}_{e,0}^{(+\infty)}} p\left(\hat{\mathbf{d}}_{e,0}^{(+\infty)} \left|\mathbf{r}\right.\right). \tag{10.1}$$

However, the computation of $p\left(\hat{\mathbf{d}}_{e,0}^{(+\infty)} \left|\mathbf{r}\right.\right)$ is very hard; in fact, it is the reason why we considered using the EM algorithm in the first place! On the other hand, if we take into account the behavior of $Q\left(\mathbf{d}_e \left|\mathbf{d}_e\right.\right)$ and the location of the solutions of the EM algorithm, a more practical decision rule pops up:

$$\hat{\mathbf{d}}_e = \arg \max_{\hat{\mathbf{d}}_{e,0}^{(+\infty)}} Q\left(\hat{\mathbf{d}}_{e,0}^{(+\infty)} \left|\hat{\mathbf{d}}_{e,0}^{(+\infty)}\right.\right). \tag{10.2}$$

The combination of multiple initial estimates with the rule (10.2) is the parallel EM algorithm. It should be noted that (10.2) is an ad-hoc criterion: from (8.31) we know that $Q\left(\mathbf{d}_e \left|\mathbf{d}_e\right.\right) \geq Q\left(\tilde{\mathbf{d}}_e \left|\tilde{\mathbf{d}}_e\right.\right)$ does not necessarily imply $p\left(\mathbf{d}_e \left|\mathbf{r}\right.\right) \geq p\left(\tilde{\mathbf{d}}_e \left|\mathbf{r}\right.\right)$.

10.4.1.1 Applications

This technique can be applied when no reliable initial estimate can be found. By choosing M_{EM} sufficiently large, and by careful selection of the initial estimates, we hope that at least one of them will lie within the acquisition region.

The parallel EM algorithm will be applied in Chapter 12 to the problems of joint phase estimation and phase ambiguity resolution, as well as joint delay estimation and frame synchronization.

10.4.1.2 Complexity

Although computational complexity in terms of *floating point operations* (flops) scales as M_{EM}, the *computation time* does not necessarily scale with M_{EM}: when the M_{EM} EM algorithms are executed on parallel processing units, computational complexity can be made independent of M_{EM}.

A further reduction in computational complexity can be achieved by performing the decision (10.2) after very few EM iterations, and then refining the best estimate with the standard EM algorithm.

10.4.2 The discrete EM algorithm

The discrete EM algorithm (D-EM) is a specialized version of the P-EM algorithm. When the space over which $\mathbf{d}_e$ is defined (say, S_{d_e}), is discrete and finite, it is well known that the standard EM algorithm may not work [111]. In particular, a situation may arise where every element $\mathbf{d}_e$ in S_{d_e} is a solution of the EM algorithm:

$$\mathbf{d}_e = \arg\max_{\mathbf{x}\in S_{d_e}} Q\left(\mathbf{x}\,|\mathbf{d}_e\,\right)$$

so that the EM algorithm converges after a single iteration.

The D-EM estimate avoids this problem by searching the best estimate in the entire $\mathbf{d}_e$-space:

$$\hat{\mathbf{d}}_e = \arg\max_{\mathbf{d}_e\in S_{d_e}} Q\left(\mathbf{d}_e\,|\mathbf{d}_e\,\right). \tag{10.3}$$

Note that, contrary to the P-EM algorithm, the discrete EM algorithm is *no longer an iterative algorithm*.

10.4.2.1 Applications

This technique can be applied to hypothesis-testing problems. This includes phase ambiguity resolution and frame synchronization, but also mode detection. Note that no initial estimate of $\mathbf{d}_e$ is required.

In Chapter 11, we will apply the discrete EM algorithm to some well known hypothesis testing problems: frame synchronization, phase ambiguity resolution and rate detection.

10.4.2.2 Complexity

The number of flops is increased with a factor equal to the size of the $\mathbf{d}_e$-space. Still, a reduction in computational complexity can be achieved by making a decision (10.3) after very few iterations of the sum-product algorithm on the synchronized factor graph. Also, complexity can be reduced further by the use of thresholds or by first creating a list of the most likely values of $\mathbf{d}_e$, based solely on a pilot sequence (a technique known as a list synchronizer) [112].

10.5 Main points

In this chapter we have exposed some of the key weaknesses of EM-based estimation algorithms. First of all, computational complexity is very high. This may be circumvented by embedding estimation iterations within the iterations of the synchronized factor graph of the detector and by performing multiple parameter updates at each iteration. Secondly, the EM algorithm suffers from severe convergence problems. Most disappointingly, the acquisition range of the EM estimator turns out to be very small. Based on a visualization of the iterative behavior of the EM algorithm, we have proposed two variations of the EM algorithm: the parallel EM algorithm, whereby multiple initial estimates are considered, and afterwards the 'best' estimate is selected. Another variation, the discrete EM algorithm, suitable for the estimation of discrete parameters, does not require an initial estimate. Again, selection of a 'best' estimate is required. Finally, based on the convergence study of the EM algorithm, we have defined a pragmatic approach to select a 'best' estimate.

Chapter 11

Estimation of discrete parameters

11.1 Introduction

Conventionally, problems such as frame synchronization and phase ambiguity resolution are solved by inserting pilot symbols in the data stream [91, 113]. As this leads to a reduction in bandwidth- and power-efficiency, many research groups have been motivated to look into code-aided hypothesis testing algorithms. The same is true to some extent for mode detection: the receiver can discover the transmit mode by inspection of the data stream and pilot symbols. Applying code-aided techniques can improve the detection performance. Note that the transmit mode could be sent from the transmitter to the receiver through a separate channel, removing the need for mode detection. Hence, the computational complexity associated with mode detection should be low. An exception to this is the problem of signal identification for military applications (i.e., eavesdropping).

In the context of frame synchronization and phase ambiguity resolution, we mention the following technical papers. Code-aided frame synchronization was discussed in [112, 114–117]: [112] uses a list-based synchronizer and makes the pilot sequence part of the codeword, thus forcing the coder into a sequence of known states. The decoder verifies this sequence to determine whether or not frame synchronization is achieved. In [114] the so-called *path surface metric*, based on the forward and backward metrics in the BCJR algorithm [33], is used for frame synchronization. The properties of this metric change when the decoder is not synchronized. In [115] termination symbols in the convolutional codes are taken into account in deriving the ML frame position. Yet another approach is mentioned in [116], where it was observed that a frame synchronization failure reduces the amplitude of the so-called extrinsic log-likelihood ratios (LLRs) as compared to a synchronized decoder. This idea was reconsidered very recently: a powerful frame synchronizer was proposed in [117], based on Mode Separation (MS): the extrinsic LLRs computed by the decoder have a bi-modal distribution. The distance between these modes is maximal when the frame is perfectly synchronized. Finally, code-aided phase ambiguity resolution was investigated in [118] where it was observed that the statistics of the branch metrics in the Viterbi decoder change depending on the phase shift. We should mention that many code-aided frame synchronization algorithms can be applied to phase ambiguity resolution with only minor modifications.

In the context of mode detection, the problem of rate detection (i.e., detecting the symbol rate $1/T$) has attracted most of the attention of the technical community. The problem of blind rate detection has already been treated extensively for DS-CDMA systems for 2nd and 3rd generation wireless devices (see for example [27,119,120] and references therein). In DS-CDMA the symbol rate is changed by fixing the chip rate and varying the length of the spreading codes (i.e., number of chip per symbol). However, the considered rate detection algorithms are very application-specific, are often developed specifically for BPSK, and, more importantly, are derived under the assumption that all channel parameters (such as channel gains, propagation delays etc.) are known. Apart from the abovementioned work for DS-CDMA, most blind rate detectors exploit the cyclostationarity of the received signal [121, 122]. Although such detectors have some very attractive properties, they fail to operate properly when the excess bandwidth decreases or when the SNR is not sufficiently high. Since the data will generally be protected by an error-correcting code, a low SNR operating point can be assumed, so that these algorithms are no longer suitable. Another type of symbol rate detector was proposed in [123]: the received signal was filtered using an analog filter bank. Through an ad-hoc criterion, the most likely signal bandwidth was determined. The authors reported an estimation accuracy of 99.5% (i.e., a rate detection error probability of 0.005).

In this chapter, we demonstrate how the discrete EM algorithm can be applied to mode detection (in particular we will focus on the important problem of rate detection) and synchronization (in particular frame synchronization and phase ambiguity resolution). We will compare the discrete EM algorithm (which accepts APPs from the *synchronized factor graph*, as described in the previous chapters) with the sum-product estimation technique, where we apply the SP

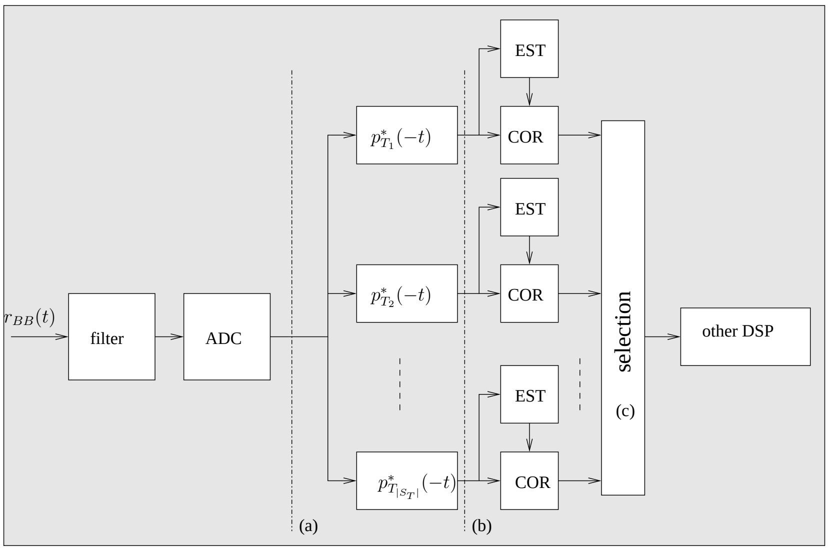


Figure 11.1: *Generic multi-rate receiver*

algorithm on the *overall factor graph* (where the synchronization parameters appear as variables (edges), as described in Chapter 9, page 125).

11.2 Mode detection

11.2.1 System model

We will demonstrate how the EM algorithm can be applied to perform mode detection. We will focus on the important problem of rate detection. The transmitted signal is given by

$$s_T(t) = \sqrt{E_s} \sum_{k=0}^{N_s-1} a_k p_T(t-kT) \tag{11.1}$$

where E_s denotes the energy per transmitted symbol, $\mathbf{a} = [a_0, \ldots, a_{N_s-1}]^T$ is the vector of data symbols and $p_T(t)$ is the transmit pulse corresponding to symbol rate $1/T$. We assume that $p_T(t)$ is a square-root Nyquist pulse. The symbol interval T belongs to a finite set of equiprobable values: $T \in S_T = \{T_{min}, \ldots, T_{max}\}$. The receiver is assumed to know the set S_T . For a flat, quasi-static channel, the complex envelope of the received signal can be expressed as

$$r_{BB}(t) = \alpha s_T(t-\tau) + n(t) \tag{11.2}$$

where $\alpha = A\exp(j\theta)$ denotes the complex channel gain, τ the propagation delay and $n(t)$ a complex AWGN process with spectral density N_0. We model the phase θ as uniformly distributed in $[0, 2\pi[$; the probability density function of the magnitude is arbitrary (e.g., in case of fading, a Rayleigh distribution is appropriate), while τ is uniformly distributed in $[-\Delta, +\Delta]$.

We consider a fully digital receiver whereby the signal $r_{BB}(t)$ is band-limited through analog filtering and sampled at a fixed rate $1/T_s$:

$$r_{BB}(kT_s) = Ae^{j\theta} s_T(kT_s - \tau) + n(kT_s) \tag{11.3}$$

with $E[n(kT_s)\, n^*(lT_s)] = N_0 T_s \delta_{k-l}$.

11.2.2 Symbol rate detection

The main goal of the receiver is to recover the data symbols. In order to do this, the receiver requires reliable estimates of A, θ, τ, and T. A generic multi-rate receiver is shown in Fig. 11.1. Its main blocks are an ADC, a matched filter bank[1] (i.e., one matched filter per symbol rate), parameter estimators and correctors. The parameter estimators (denoted by

[1] In practice this bank of matched filters could be replaced by a single matched filter, preceded by an interpolator, as explained in Chapter 7.

EST) estimate τ, θ and A, while the parameter correctors (denoted by COR), perform compensation for carrier phase, amplitude and delay. The latter operation, which includes timing correction and possibly sample rate conversion, is performed by a digital interpolator. From Chapter 9 we know that the first step in the estimation process is timing recovery. As timing recovery requires knowledge of T, rate detection should ideally take place prior to any other estimation algorithms. We will consider three rate detection algorithms.

11.2.2.1 Cyclic correlation-based algorithm

A cyclic correlation-based algorithm from [121] performs rate detection in front of the matched filter bank. The branch corresponding to the estimated rate is selected. Such an approach, exploiting the cyclo-stationary character of the incoming signal, has the advantage of very low complexity. This algorithm corresponds to performing rate detection at point (a) in Fig. 11.1.

The idea is as follows: in order to eliminate excess noise, we first apply the signal samples $\{r_{BB}(kT_s)\}$ to a low-pass filter. This results in a sequence of N samples, denoted by $\mathbf{z}$. We then define, for some $\Upsilon \in \mathbb{N}$:

$$\begin{aligned} \mathbf{y}_2(n) &\doteq [z(n-\Upsilon) z^*(n), \ldots, z(n+\Upsilon) z^*(n)] && (11.4) \\ \mathbf{r}_N(T) &\doteq \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}_2(n) e^{-j2\pi n T_s/T}. && (11.5) \end{aligned}$$

Here $\mathbf{r}_N(T)$ measures the spectral line at $f = 1/T$. Note that a signal with symbol rate $1/T$ should yield spectral lines in the FT of its autocorrelation at frequencies, $f \in \{0, -1/T, +1/T\}$. The final rate detection algorithm is given by

$$\hat{T} = \arg\max_T \|\mathbf{r}_N(T)\|^2. \quad (11.6)$$

The parameter Υ will be set to 0 for the remainder of this text.

11.2.2.2 Low-SNR approximation

A second algorithm is based on a low-SNR approximation of the likelihood function [124]. This algorithm requires no knowledge of τ, A or θ.

This algorithm, the details of which can be found in the Appendix of this chapter (section 11.6), can be interpreted as selecting the branch in the matched filter bank that yields the largest output energy. It is computationally more complex than the cyclic correlation approach from [121], since now the incoming signal has to be filtered by each of the matched filters. Only after considering the outputs of the matched filter bank, a decision is made w.r.t. the symbol rate. This corresponds to performing rate detection at point (b) in Fig. 11.1.

11.2.2.3 Discrete EM

Applying the discrete EM algorithm leads to the following E-step, with $\mathbf{d}_e = T$:

$$Q(T|T) = \Re\left\{\hat{\alpha}\sum_{k=0}^{N_s-1} \widetilde{a}_k y_T^*(kT+\hat{\tau})\right\} \quad (11.7)$$

where $y_T(t) = T_s \sum_k r_{BB}(kT_s) p_T^*(t-kT_s)/\sqrt{E_s}$ and $\widetilde{a}_k$ is the familiar a posteriori symbol expectation of the k-th symbol:

$$\widetilde{a}_k = \sum_{\omega\in\Omega} \omega \times p(a_k = \omega| T, \mathbf{r}, \hat{\tau}, \hat{\alpha}) \quad (11.8)$$

and $\hat{\alpha}$ and $\hat{\tau}$ are estimates of the complex gain and the propagation delay, respectively. For fixed T, finding these estimates can be achieved through the techniques described in Chapter 9. Hence, the symbol rate estimate is given by

$$\hat{T} = \arg\max_{T\in S_T} Q(T|T). \quad (11.9)$$

In the case of uncoded transmission, computation of the a posteriori symbol probabilities is a simple task:

$$p(a_k = \omega| T, \mathbf{r}, \hat{\tau}, \hat{\alpha}) = C\exp\left(-\frac{1}{N_0}|y_T(kT+\hat{\tau}) - \hat{\alpha}\omega|^2\right) \quad (11.10)$$

with C a normalizing constant. In the case of coded transmission, the a posteriori symbol probabilities are computed by the sum-product algorithm on the synchronized factor graph. As the latter approach will drastically increase the overall computational complexity, it is preferred to treat the data symbols as uncoded during the rate detection process: this is achieved by simply removing the part corresponding to the code in the synchronized factor graph during rate detection.

From (11.7), we see that the received signal $r_{BB}(t)$ is applied to the matched filter bank, and to each filter output, an algorithm for estimating τ, A and θ is applied. This estimation can be performed by means of a classical algorithm (see [16, 17, 21]) or by means of a more sophisticated EM algorithm (see [19, 98, 125]). Hence, for each branch in the matched filter bank there are corresponding estimates of τ, A and θ. These estimates are used to compute the marginal APPs required by the discrete EM algorithm. When we treat the data as uncoded during rate detection, this corresponds to performing rate detection at point (c) in Fig. 11.1.

A similar line of reasoning can be applied to detect any type of mode, such as the signaling constellation, the code rate, etc. In some cases it will be necessary to exploit information from the code, possibly leading to fairly complex algorithms.

Performance results are postponed until Chapter 12.

11.3 Code-aided estimation of discrete synchronization parameters

Code-aided frame synchronization and phase ambiguity resolution can be performed either on the *synchronized factor graph* (e.g., using the discrete EM algorithm), or on the *overall factor graph* (whereby the delay shift and phase ambiguity are considered as variables (edges) in the factor graph). This latter technique was discussed in section 9.4, where we have shown that for continuous parameters, the sum-product algorithm runs into some practical difficulties. When the synchronization parameters are discrete, these problems are removed and the sum-product algorithm again becomes an attractive way to perform joint detection and estimation.

11.3.1 Phase ambiguity resolution

11.3.1.1 System model

We start from the following model: a sequence $\mathbf{a}$ in Ω^{N_s} is transmitted, rotated over θ and corrupted by AWGN:

$$\begin{aligned} \mathbf{r} &= \mathbf{a}e^{j\theta} + \mathbf{n} && (11.11) \\ &= \mathbf{x} + \mathbf{n} && (11.12) \end{aligned}$$

where $E\left[\mathbf{n}\mathbf{n}^H\right] = 2\sigma^2\mathbf{I}_{N_s}$. The phase θ is constrained to the following set of (equiprobable) values $S_\theta = \{0, 2\pi/M_\Omega, \ldots, 2\pi\,(M_\Omega$ where M_Ω is the ambiguity number of the constellation (as defined in section 9.2.2). We remind that for M-PSK constellations, $M_\Omega = M$, where M is the number of constellation points. For square QAM constellations, $M_\Omega = 4$.

11.3.1.2 ML approach

According to the ML criterion, we obtain an estimate of the data sequence as:

$$\begin{aligned} \hat{\mathbf{a}} &= \arg\max_{\mathbf{a}} p\left(\mathbf{r}|\,\mathbf{a}\right) && (11.13) \\ &= \arg\max_{\mathbf{a}} \sum_{\theta} p\left(\mathbf{r}|\,\mathbf{a}, \theta\right). && (11.14) \end{aligned}$$

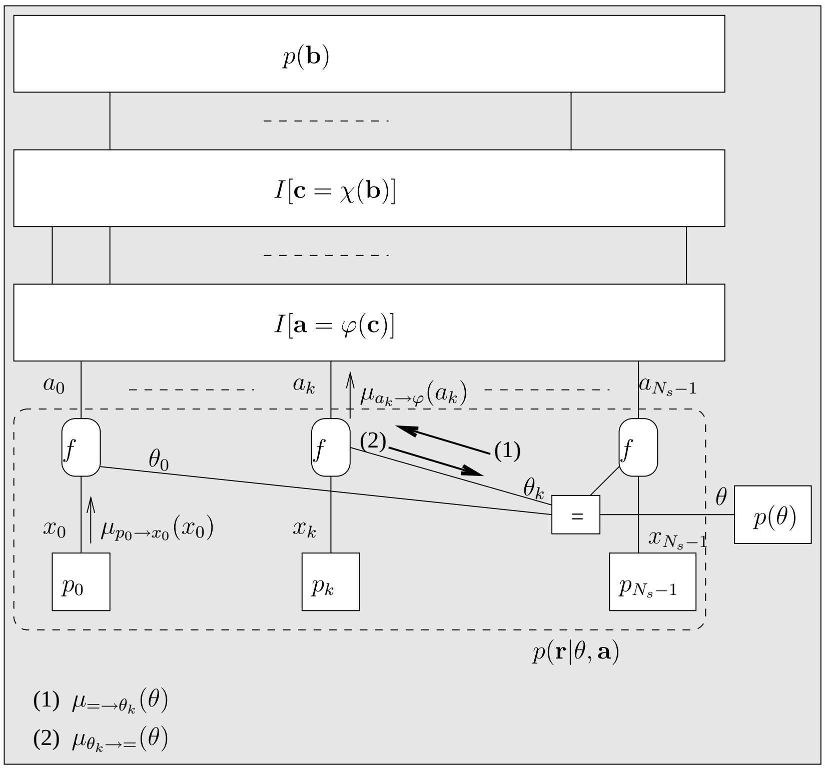
The computation (11.13) is intractable in practice.

11.3.1.3 EM approach

Applying the discrete EM algorithm leads to

$$\hat{\theta} = \arg\max_{\theta} \Re\left\{\mathbf{r}^H\tilde{\mathbf{a}}e^{j\theta}\right\} \qquad (11.15)$$

where $\tilde{\mathbf{a}} = \left[\tilde{a}_0, \ldots, \tilde{a}_{N_s-1}\right]^T$ and $\tilde{a}_k = E\left[a_k\,|\mathbf{r}, \theta\right]$. The required APPs are computed through the sum-product algorithm on the synchronized factor graph.



$p(\mathbf{b})$
$I[\mathbf{c} = \chi(\mathbf{b})]$
$I[\mathbf{a} = \varphi(\mathbf{c})]$
a_0
a_k
$\mu_{a_k \to \varphi}(a_k)$
a_{N_s-1}
f
f
f
(2)
(1)
θ_0
θ_k
θ
=
$p(\theta)$
x_0
$\mu_{p_0 \to x_0}(x_0)$
x_k
x_{N_s-1}
p_0
p_k
p_{N_s-1}
$p(\mathbf{r}|\theta, \mathbf{a})$
(1) $\mu_{= \to \theta_k}(\theta)$
(2) $\mu_{\theta_k \to =}(\theta)$

Figure 11.2: *Overall factor graph with phase ambiguity. The f-nodes enforce $f(a_k, x_k, \theta_k) = I\left[x_k = a_k e^{j\theta_k}\right]$.*

11.3.1.4 Overall factor-graph approach

The overall factor graph corresponding to $p(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \theta\,|\mathbf{r})$ is shown in Fig. 11.2. This is an exact copy of Fig. 9.4, with the important difference that in Fig. 11.2 the phase can only take on discrete values. Replacing in the sum-product algorithm from page 124, integrals with summations, at the first iteration, we obtain

$$\mu_{\boxminus \to \theta_k}(\theta_k) = \frac{1}{M_\Omega} \tag{11.16}$$

so that the messages to the mapper nodes are given by

$$\begin{aligned} \mu_{a_k \to \varphi}(a_k) &= \sum_{\theta_k \in S_\theta} \mu_{p_k \to x_k}\left(a_k e^{j\theta_k}\right) \mu_{\boxminus \to \theta_k}(\theta_k) && (11.17) \\ &= \frac{1}{M_\Omega} \sum_{\theta_k \in S_\theta} \mu_{p_k \to x_k}\left(a_k e^{j\theta_k}\right) && (11.18) \end{aligned}$$

Due to the symmetries in the constellations,

$$\mu_{f \to a_k}(a_k) = \mu_{f \to a_k}\left(a_k e^{j\theta}\right) \tag{11.19}$$

for any $\theta \in S_\theta$. Hence, $\mu_{f \to a_k}(a_k)$ can only take on $|\Omega|/M_\Omega$ different values. Let us focus on the interesting case of M-PSK[2] signaling: then $|\Omega|/M_\Omega = 1$, so that $\mu_{f \to a_k}(a_k = \omega) = 1/M, \forall \omega \in \Omega$. This means that no useful information is sent over the a_k edges to the mapper nodes. The block $I[\mathbf{a} = \varphi(\mathbf{c})]$ is not provided with any useful information by the messages $\mu_{f \to a_k}(a_k)$. It can be shown that for most practical codes and mapping strategies, after application of the SP algorithm in the nodes $I[\mathbf{a} = \varphi(\mathbf{c})]$ and $I[\mathbf{c} = \chi(\mathbf{b})]$, the downward messages $\mu_{a_k \to f}(a_k)$ will also be uniform (and hence contain no useful information). This causes the sum-product algorithm on the overall factor graph to converge after a single iteration! Even for general codes and mapping strategies, there is no guarantee that the sum-product algorithm will always converge to the correct a posteriori probabilities (due to the cycles in the overall factor graph). Furthermore, it may take many iterations for the sum-product algorithm to converge.

[2]With $\Omega = \{\exp(j2\pi k/M)\}_{k=0}^{M-1}$.

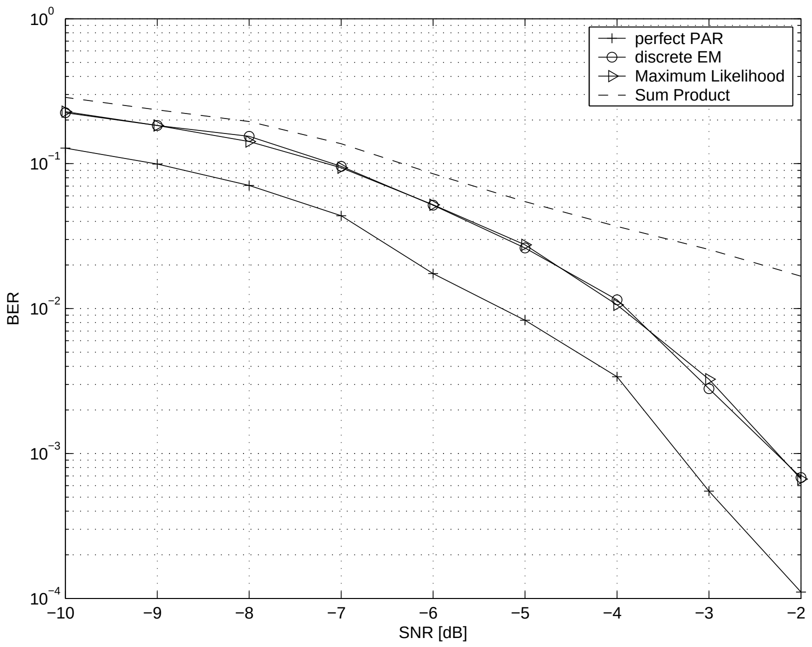


Figure 11.3: *Phase Ambiguity resolution - 16-PSK - random code*

11.3.1.5 Performance results

To compare these algorithms, we have carried out computer simulations for a short random code, consisting of 16 codewords of 120 bits, mapped to a 16-PSK constellation. To remove the effect of cycles within and between the nodes $I[\mathbf{a} = \varphi(\mathbf{c})]$ and $I[\mathbf{c} = \chi(\mathbf{b})]$, we create a cycle-less factor graph of the function $I[\mathbf{a} = \varphi(\chi(\mathbf{b}))]$, without further factorization. Hence, the SP algorithm within this part of the graph is exact. We have considered three detectors:

1. the ML detector which performs data detection according to (11.13)
2. the discrete EM estimator in combination with the synchronized factor graph
3. the sum-product algorithm on the overall factor graph

In Fig. 11.3 we show bit-error-rate (BER) results for phase ambiguity resolution. Clearly the discrete EM algorithm outperforms the sum-product algorithm by a wide margin. Most strikingly, the discrete EM estimator has nearly the same performance as the ML detector. Note also that there remains a gap between the performance of the ML detector and the BER of a perfectly synchronized system: this degradation is related to the inherent properties of the code and mapping and cannot be bridged by any estimation algorithm. We have also evaluated the performance of a similar system with 16-QAM signaling, and the results (not shown) are roughly the same.

11.3.2 Frame synchronization

11.3.2.1 System model

Now, the model is a little more complex. Let us start again from a discrete-time system where the transmitted sequence is delay-shifted over k_τ symbol durations and corrupted with AWGN. We assume the delay shift to be in the set $k_\tau \in \{0, \ldots, M_\tau - 1\}$. This yields the following model:

$$\begin{aligned} \mathbf{r} &= \left[\mathbf{0}_{k_\tau}^T \, \mathbf{a}^T \, \mathbf{0}_{M_\tau-1-k_\tau}^T\right]^T + \mathbf{n} && (11.20)\\ &= D^{k_\tau}(\mathbf{a}) + \mathbf{n} && (11.21)\\ &= \mathbf{x} + \mathbf{n} && (11.22) \end{aligned}$$

where $\mathbf{0}_m$ is a vector consisting of m zeros, $E\left[\mathbf{n}\mathbf{n}^H\right] = 2\sigma^2\mathbf{I}$, and $D^k(.)$ is an invertible function $D^k : \Omega^{N_s} \rightarrow \{\Omega \cup \{0\}\}^{N_s+M_\tau-1}$, which, for $k \in \{0, \ldots, M_\tau - 1\}$, prefixes to its argument k zeros and postfixes $M_\tau - 1 - k$ zeros. The inverse of this function will be denoted by $D^{-k}(.)$, so that $D^{-k}\left(D^k(\mathbf{a})\right) = \mathbf{a}$.

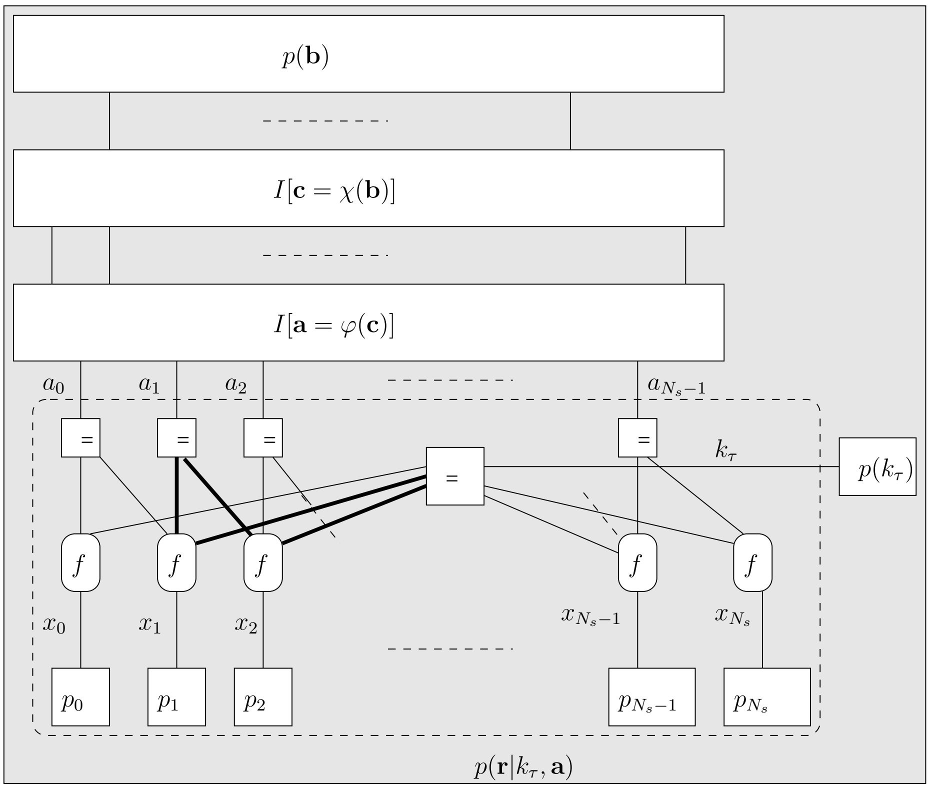


Figure 11.4: *Overall factor graph with delay shift k_τ. A cycle of length 4 is shown in bold.*

11.3.2.2 EM approach

Applying the discrete EM algorithm leads to

$$\hat{k}_\tau = \arg\max_{k_\tau} \Re\left\{\mathbf{r}^H D^{k_\tau}(\tilde{\mathbf{a}})\right\} - \frac{1}{2}\sum_{k=0}^{N_s-1} \widetilde{|a_k|^2}. \tag{11.23}$$

where $\tilde{\mathbf{a}} = [\tilde{a}_0, \ldots, \tilde{a}_{N_s-1}]$, $\tilde{a}_k = E[a_k|\mathbf{r}, k_\tau]$ and $\widetilde{|a_k|^2} = E\left[|a_k|^2|\mathbf{r},\tau\right]$.

11.3.2.3 Overall factor graph approach

A factor graph of the factorization of $p(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{x},k_\tau|\mathbf{r})$ is depicted in Fig. 11.4 for $M_\tau = 2$. The nodes marked by f correspond to the following constraint:

$$f(a_k, a_{k-1}, x_k, k_\tau) = I[(a_k = x_k \wedge k_\tau = 0) \vee (a_{k-1} = x_k \wedge k_\tau = 1)] \tag{11.24}$$

where $\wedge$ ($\vee$) denotes the logical 'and' ('or'). Application of the sum-product algorithm on the overall factor graph is straightforward and omitted here.

Contrary to the problem of phase ambiguity resolution, the overall factor graph from Fig. 11.4 has one important weakness: the node corresponding to $p(\mathbf{r}|k_\tau,\mathbf{a})$ contains many very short cycles (of length 4). Such cycles will seriously degrade the performance of the sum-product algorithm, especially when they occur in great number [8,34, 126].

11.3.2.4 Performance results

To compare the performance of the frame synchronization techniques, we have again carried out computer simulations, for the same code as in section 11.3.1.5. We set $M_\tau = 3$ and consider three detectors:

1. the ML detector which performs data detection according[3] to (11.13)

[3]With θ replaced by k_τ, of course.

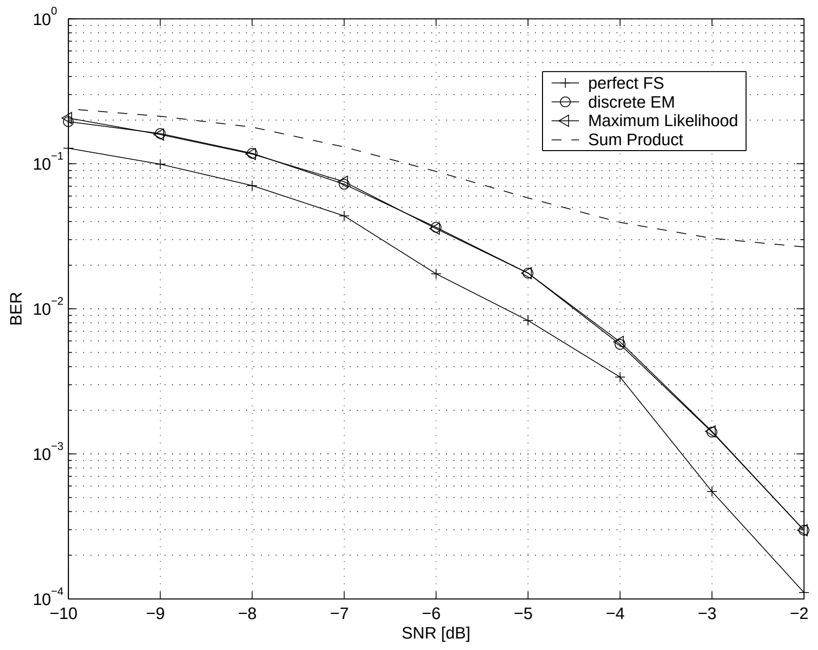


Figure 11.5: *Frame Synchronization - 16-PSK - random code*

2. the discrete EM estimator in combination with the synchronized factor graph

3. the sum-product algorithm on the overall factor graph

In Fig. 11.5 we see that the sum-product algorithm is again outperformed by the discrete EM algorithm, which in turn attains near-ML performance. In light of this evidence, the sum-product algorithm may not be the best tool for performing detection in the presence of unknown discrete parameters.

11.4 Other code-aided hypothesis testing algorithms

We briefly describe two hypothesis-testing algorithms from technical literature that operate on the *synchronized factor graph*, much like the discrete EM algorithm. We refer to Fig. 11.6 for the relevant notations. These algorithms will be applied in Chapter 12.

Mode Separation

The first algorithm (Mode Separation [117]) is based on the following observation: the messages that are computed by the decoder $\mu_{c_k \to \varphi}(c_k)$ will have a different distribution depending on whether or not the receiver is synchronized. The technique is described in Algorithm 6. Note that messages are first converted to log-likelihood ratios.

Algorithm 6 Mode Separation

```
1: for $k_\tau = 0$ to $M_\tau - 1$ do
2:   for $k_\theta = 0$ to $M_\Omega - 1$ do
3:     $\mathbf{y} = D^{-k_\tau}(\mathbf{r})\, e^{-jk_\theta 2\pi/M_\Omega}$
4:     perform SP algorithm:
         *compute $\mu_{a_k \to \varphi}(a_k) \propto p(y_k | a_k)$
         *compute $\mu_{\varphi \to c_k}(c_k)$
         *decode and compute $\mu_{c_k \to \varphi}(c_k)$
5:     estimate Mode Separation
         *$\lambda_k = \log(\mu_{c_k \to \varphi}(c_k = 1)) - \log(\mu_{c_k \to \varphi}(c_k = 0))$
         *$M_+ = E[\lambda | \lambda > 0]$
         *$M_- = E[\lambda | \lambda < 0]$
         *$MS(k_\tau, k_\theta) = M_+ - M_-$
6:   end for
7: end for
8: $\left[\hat{k}_\tau, \hat{k}_\theta\right] = \arg\max_{k_\tau, k_\theta} MS(k_\tau, k_\theta)$
```

Pseudo-ML

A second ad-hoc technique (pseudo-ML [127]) is described in Algorithm 7.

Algorithm 7 Pseudo-ML

```
1: for $k_\tau = 0$ to $M_\tau - 1$ do
2:   for $k_\theta = 0$ to $M_\Omega - 1$ do
3:     $\mathbf{y} = D^{-k_\tau}(\mathbf{r})\, e^{-jk_\theta 2\pi/M_\Omega}$
4:     perform SP algorithm:
         *compute $\mu_{a_k \to \varphi}(a_k) \propto p(y_k | a_k)$
         *compute $\mu_{\varphi \to c_k}(c_k)$
         *decode and compute $\mu_{c_k \to \varphi}(c_k)$ and $\mu_{a_k \to \varphi}(a_k)$
5:     compute pseudo-likelihood
         *$\lambda_k = \log\left(\sum_{\omega \in \Omega} \mu_{a_k \to \varphi}(a_k = \omega) \times \mu_{a_k \to \varphi}(a_k = \omega)\right)$
         *$PML(k_\tau, k_\theta) = \sum_k \lambda_k$
6:   end for
7: end for
8: $\left[\hat{k}_{\tau,MS}, \hat{k}_{\theta,MS}\right] = \arg\max_{k_\tau, k_\theta} PML(k_\tau, k_\theta)$
```

11.5 Main points

In this chapter, we have described two hypothesis testing algorithms. The first is the discrete EM algorithm which accepts a posteriori probabilities from the synchronized factor graph. The second is the sum-product algorithm on the overall factor graph. In the discrete EM algorithm the different hypotheses are tested separately (possibly in parallel), while the SP algorithm on the overall factor graph can be interpreted as testing all hypotheses simultaneously.

We have considered the following hypothesis testing problems: symbol rate detection, frame synchronization and phase ambiguity resolution. We have argued that for practical systems, applying the SP algorithm on the overall factor graph may not be the best choice: for both phase ambiguity resolution and frame synchronization, cycles in the overall factor graph cause the sum-product algorithm to converge to incorrect APPs. Through computer simulations we have shown that, for the same two problems, the discrete EM algorithm systematically outperforms the sum-product algorithm on the overall factor graph, and even achieves near-ML performance.

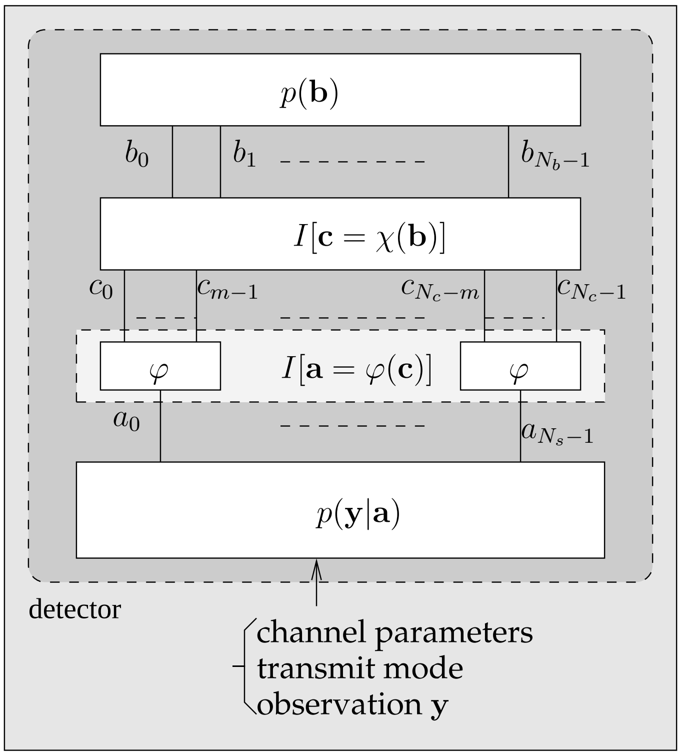


Figure 11.6: *Data detection: factor graph of* $p(\mathbf{b})\, I[\mathbf{c} = \chi(\mathbf{b})]\, I[\mathbf{a} = \varphi(\mathbf{c})]\, p(\mathbf{y}|\, \mathbf{a})$. *The observation* $\mathbf{y}$*, the channel parameters and transmit mode are* parameters *(not variables) in this graph.*

11.6 Appendix: Rate detection - low-SNR method

We denote by $\mathbf{r}$ the expansion of $r_{BB}(t)$ onto a suitable basis. The ML estimate of T is obtained by maximizing the likelihood function [17]:

$$\hat{T}_{ML} = \arg \max_{T \in S_T} p(\mathbf{r}\,|T) \tag{11.25}$$

with

$$p(\mathbf{r}\,|T) = E_{\mathbf{a},\tau,\alpha}\left[p(\mathbf{r}\,|\mathbf{a}, \tau, \alpha, T)\right] \tag{11.26}$$

where $E_{\mathbf{a},\tau,\alpha}[.]$ denotes the averaging over all possible data sequences, τ and α. We model the phase θ as uniformly distributed in $[0, 2\pi[$; the probability density function of the magnitude is arbitrary (e.g., in case of fading, a Rayleigh distribution is appropriate), while τ is uniformly distributed in $[-\Delta, +\Delta]$. Taking into account the AWGN noise, we can write

$$\begin{aligned} & p(\mathbf{r}\,|\mathbf{a}, \tau, \alpha, T) \\ & = C \exp\left(-\frac{1}{N_0}\sum_k \left|r_{BB}(kT_s) - \alpha s_T(kT_s - \tau)\right|^2\right) && (11.27) \\ & = C' \exp\left(\frac{2E_s}{N_0}\Re\left\{\alpha^* \sum_{k=0}^{N_s-1} a_k^* y_T(kT + \tau)\right\}\right) && (11.28) \end{aligned}$$

where C and C' do not depend on T and $y_T(t) = T_s \sum_k r_{BB}(kT_s)\, p_T^*(t - kT_s)/\sqrt{E_s}$. The quantities $y_T(kT + \tau)$ are obtained by applying the received signal to a filter matched to the transmit pulse, corresponding to symbol rate $1/T$. Unfortunately, averaging in (11.26) w.r.t. the unknown data symbols is generally intractable. Let us expand $p(\mathbf{r}\,|\mathbf{a}, \tau, \alpha, T)$ in a Taylor series. We start from $p(\mathbf{r}\,|T) = E_{\mathbf{a},\tau,\alpha}[p(\mathbf{r}\,|\mathbf{a}, \tau, \alpha, T)]$ with (up to an irrelevant multiplicative constant):

$$p(\mathbf{r}\,|\mathbf{a}, \tau, \alpha, T) = \exp\left(\frac{2E_s}{N_0}\Re\left\{\sum_{k=0}^{N_s-1} z_k\right\}\right) \tag{11.29}$$

where we have introduced $z_k \doteq \alpha^* a_k^* y_T\left(kT+\tau\right)$. Expansion in a Taylor series yields

$$\begin{aligned} p\left(\mathbf{r}\left|\mathbf{a},\tau,\alpha,T\right.\right) &\approx& 1+\frac{2E_s}{N_0}\Re\left\{\sum_{k=0}^{N_s-1} z_k\right\} \\ && +\frac{1}{2}\left(\frac{2E_s}{N_0}\right)^2\left(\Re\left\{\sum_{k=0}^{N_s-1} z_k\right\}\right)^2 \end{aligned} \qquad (11.30)$$

The first term in (11.30) is independent of T and can be dropped. The second term is linear in a_k, so that, since $E\left[a_k\right]=0$, $E\left[z_k\right]=0$. Within irrelevant constants, this results in

$$\begin{aligned} p\left(\mathbf{r}\left|\tau,\alpha,T\right.\right) &=& E_{\mathbf{a}}\left[\left(\Re\left\{\sum_{k=0}^{N_s-1} z_k\right\}\right)^2\right] & \qquad (11.31)\\ &\propto& E_{\mathbf{a}}\left[\sum_{k,k'}\left(z_k+z_k^*\right)\left(z_{k'}+z_{k'}^*\right)\right] & \qquad (11.32)\\ &=& \sum_k E_{\mathbf{a}}\left[\left(z_k+z_k^*\right)^2\right] & \qquad (11.33)\end{aligned}$$

since $E_{\mathbf{a}}\left[z_k z_{k'}^*\right]=0$ and $E_{\mathbf{a}}\left[z_k z_{k'}\right]=0$ when $k\neq k'$. We now evaluate $E_{\mathbf{a}}\left[\left(z_k+z_k^*\right)^2\right]$, abbreviating $y_T\left(kT+\tau\right)$ with y_k:

$$\begin{aligned} & E_{\mathbf{a}}\left[\left(z_k+z_k^*\right)^2\right]= \\ & \alpha^2\left(y_k^*\right)^2 E_{\mathbf{a}}\left[a_k^2\right]+\left(\alpha^*\right)^2 y_k^2 E_{\mathbf{a}}\left[\left(a_k^*\right)^2\right]+\left|\alpha\right|^2\left|y_k\right|^2 E_{\mathbf{a}}\left[\left|a_k\right|^2\right] \end{aligned} \qquad (11.34)$$

It can easily been seen that $E_{\mathbf{a}}\left[\left|a_k\right|^2\right]=1$, $E_{\mathbf{a}}\left[a_k^2\right]=E_{\mathbf{a}}\left[\left(a_k^*\right)^2\right]=C$ where $C=0$ for complex constellations and $C=1$ for real constellations. Additionally, since $\alpha=Ae^{j\theta}$ with θ uniformly distributed in $[0,2\pi[$, $E_\theta\left[\alpha^2\right]=E_\theta\left[\left(\alpha^*\right)^2\right]=0$, so that averaging over θ of (11.33) yields (irrespective of the distribution of A):

$$p\left(\mathbf{r}\left|\tau,T\right.\right)\propto\sum_{k=0}^{N_s-1}\left|y_T\left(kT+\tau\right)\right|^2. \qquad (11.35)$$

Averaging over τ yields:

$$\begin{aligned} p\left(\mathbf{r}\left|T\right.\right) &\propto& \sum_{k=0}^{N_s-1}\int_{-\Delta}^{+\Delta}\left|y_T\left(kT+\tau\right)\right|^2 d\tau & \qquad (11.36)\\ &=& \int_{-\Delta}^{(N_s-1)T+\Delta} w\left(u\right)\left|y_T\left(u\right)\right|^2 du & \qquad (11.37)\end{aligned}$$

where $w\left(u\right)$ is a window function that depends on N_s, Δ and T. Assuming $N_sT\geq 2\Delta$, $w\left(u\right)$ is constant and proportional to $1/T$ in the interval $\Delta\leq u\leq N_sT-\Delta$. For $\Delta/T\ll N_s$, we can approximate (11.37) with

$$\begin{aligned} p\left(\mathbf{r}\left|T\right.\right) &\propto& \frac{1}{T}\int_{-\Delta}^{(N_s-1)T+\Delta}\left|y_T\left(u\right)\right|^2 du & \qquad (11.38)\\ &\approx& \frac{T_s}{T}\sum_{k=0}^{\lceil((N_s-1)T+\Delta)/T_s\rceil}\left|y_T\left(kT_s\right)\right|^2. & \qquad (11.39)\end{aligned}$$

Substituting (11.39) into (11.25) yields the final symbol rate detection algorithm. This algorithm can be interpreted as selecting the branch in the matched filter bank that yields the largest output energy.

Chapter 12

Code-aided estimation: performance results

12.1 Introduction

Now that we have given a detailed description of the application of the EM and SAGE algorithms to channel estimation, synchronization, mode detection, frame synchronization and phase ambiguity resolution, we will present some simulation results. Before we continue, note that many of the examples below are chosen especially to show off the power of EM-based estimation. In some cases, more conventional algorithms will perform satisfactorily. In other cases, the EM-based estimation can be applied without use of the error-correcting code. So, the choice of estimation algorithms should be made on a case-by-case basis.

This chapter is organized as follows. We first define the performance measures we will consider to compare different estimation algorithms in section 12.2. We then move on to practical estimation problems: first the frequency-flat channel is investigated in considerable detail in section 12.3. This is followed by a more concise look into frequency-selective channels in section 12.4. We end with the application of the discrete EM algorithm to rate detection in section 12.5.

12.2 Performance measure

12.2.1 Measures

The ultimate performance criterion is without any doubt the Frame-Error-Rate (FER, also known as Packet-Error-Rate), i.e., the fraction of packets that could not be recovered correctly. For some applications, the Bit-Error-Rate (BER) is also a useful performance measure. Finally, we will also consider the Mean Square Estimation Error (MSEE), i.e., the variance of the estimation error. Note that these performance measures are not equivalent: two algorithms may yield very different MSEE, but achieve the same BER performance (and the other way round). Similarly, different algorithms may give rise to different BER performance but the same FER performance (and the other way round). We mention this to warn the reader that an impressive gain in terms of one performance measure does not necessarily correspond to an equally impressive gain in terms of another performance measure.

12.2.2 Benchmarks

The FER and BER performance of estimation algorithms will be compared to the FER and BER performance of a receiver with perfect knowledge of all unknown parameters (a *genie* receiver[1]). The MSEE performance of estimation algorithms is benchmarked against the Modified Cramer-Rao Bound (MCRB) [128]. For unbiased estimators, the MCRB is a lower bound on the MSEE: no algorithm can achieve an MSEE lower than the MCRB. The MCRB generally becomes tighter with increasing SNR.

12.3 Channel estimation: frequency-flat channel

12.3.1 System set-up

We will consider a turbo code where the constituent convolutional encoders are recursive and systematic with octal generators $(21, 37)_8$ and constraint length $\nu = 5$. Only the first convolutional code is terminated. The interleaver is

[1]A genie provides the receiver with all unknown parameters. Also: a genie-aided receiver.

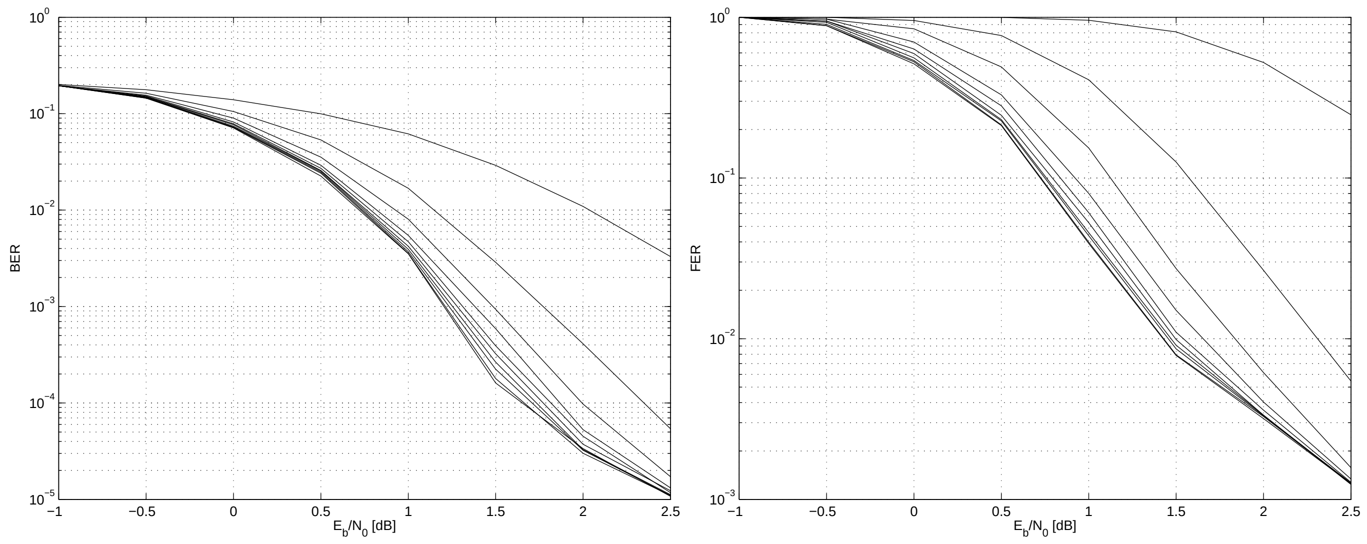


Figure 12.1: *BER (left) and FER (right) performance of genie receiver. Different curves correspond to different iterations (from one to ten).*

pseudo-random, varies from frame to frame and has length $N_b = 240$ bits. The turbo code has a rate $1/3$, yielding $N_c = 720$ coded bits. These bits are interleaved prior to Gray-mapping onto a 4-PSK constellation. This results in a scheme known as bit-interleaved coded modulation (BICM) [35,129,130]. The $N_s = 360$ 4-PSK symbols are placed on square-root cosine roll-off pulses with roll-off factor 0.2 (20%). The resulting signal is transmitted over an equivalent complex baseband channel

$$h_{BB}\left(t\right) = e^{j\theta}\delta\left(t-\tau\right). \tag{12.1}$$

This means we consider the channel amplitude to be known. The received complex baseband signal is sampled at four times the symbol rate. After estimation of θ and τ, matched filter outputs are timing-corrected and rotated, yielding $\mathbf{y} = \left[y_0, \ldots, y_{N_s-1}\right]^T$, which is provided as a parameter to a synchronized factor graph, representing the factorization of $p\left(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}\left|\mathbf{r}, \hat{\tau}, \hat{\theta}\right.\right) = p\left(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}\left|\mathbf{y}\right.\right)$, where $\mathbf{x}$ is a vector of additional variables and where $\hat{\tau}$ and $\hat{\theta}$ are the estimates of the propagation delay and carrier phase, respectively. Applying the sum-product algorithm on this graph results in (approximations of) the a posteriori probabilities $\left\{p\left(a_k\left|\mathbf{r}, \hat{\tau}, \hat{\theta}\right.\right)\right\}$, $\left\{p\left(b_k\left|\mathbf{r}, \hat{\tau}, \hat{\theta}\right.\right)\right\}$ and $\left\{p\left(c_k\left|\mathbf{r}, \hat{\tau}, \hat{\theta}\right.\right)\right\}$. Due to the cycles in the graph, the sum-product algorithm will be iterative. We will perform 10 decoding iterations in the synchronized factor graph at the receiver.

Estimation

All code-aided estimation algorithms are *embedded* algorithms (as described in Chapter 10), whereby we update the parameter estimates after each decoding iteration, without resetting the synchronized factor graph. More details w.r.t. the exact implementation will be provided as we go along.

12.3.2 Genie receiver

Let us first evaluate the performance of the genie receiver, which has perfect knowledge of both θ and τ. In Fig. 12.1 we show BER and FER as a function of the SNR (expressed in E_b/N_0 in dB) for different iterations.

12.3.3 Phase estimation

In a first phase, we will assume that τ is perfectly known to the receiver.

Fractional phase estimation

Consider the problem where the carrier phase ambiguity has been resolved. Hence, the (initial) estimation error is in the interval $[-\pi/4, +\pi/4]$. Three estimation algorithms will be investigated:

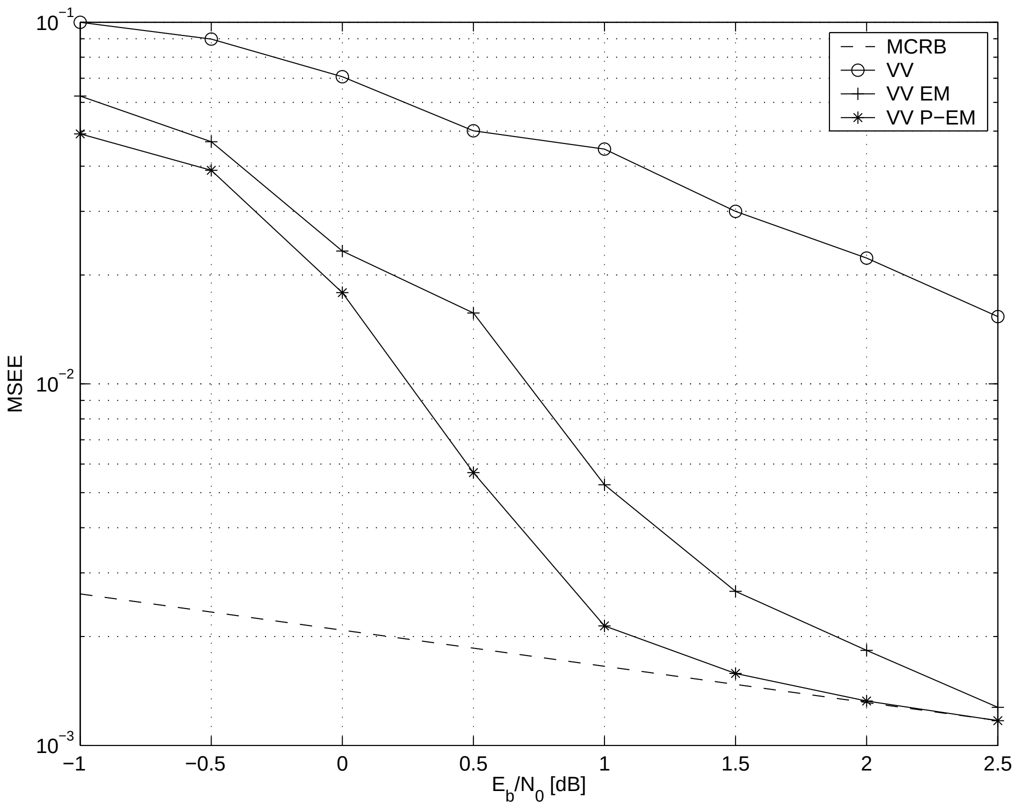


Figure 12.2: *Phase estimation: MSEE performance for fractional phase estimation after 10 EM iterations.*

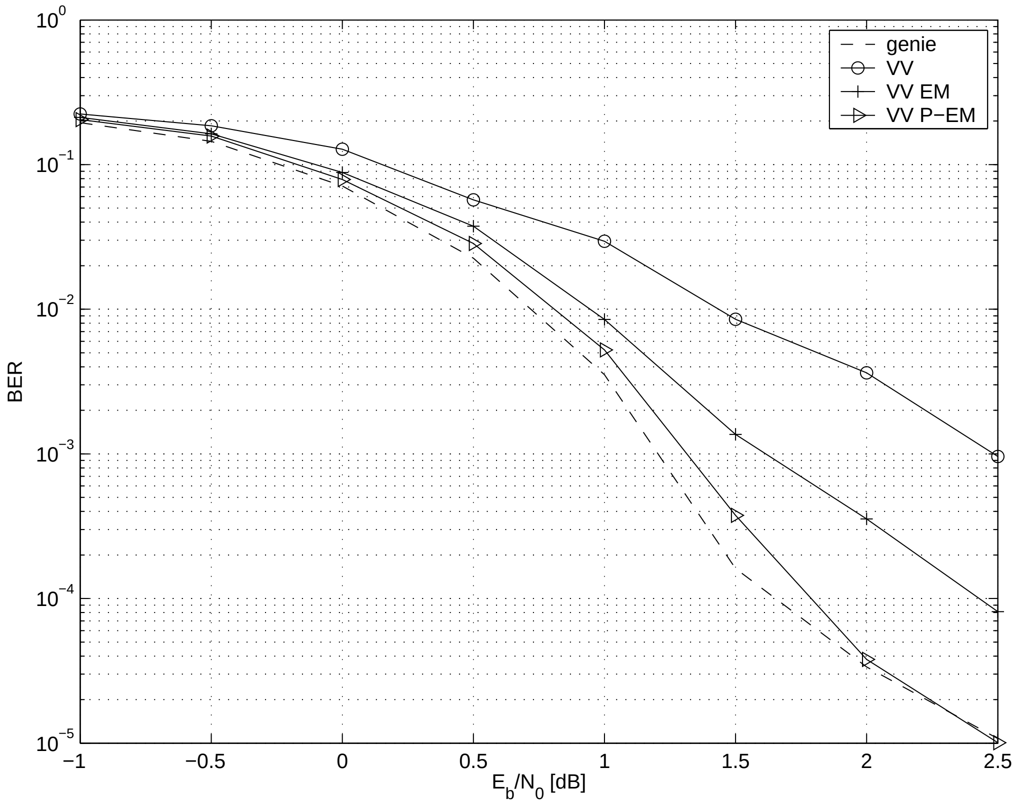


Figure 12.3: *Phase estimation: BER performance for fractional phase estimation after 10 EM iterations.*

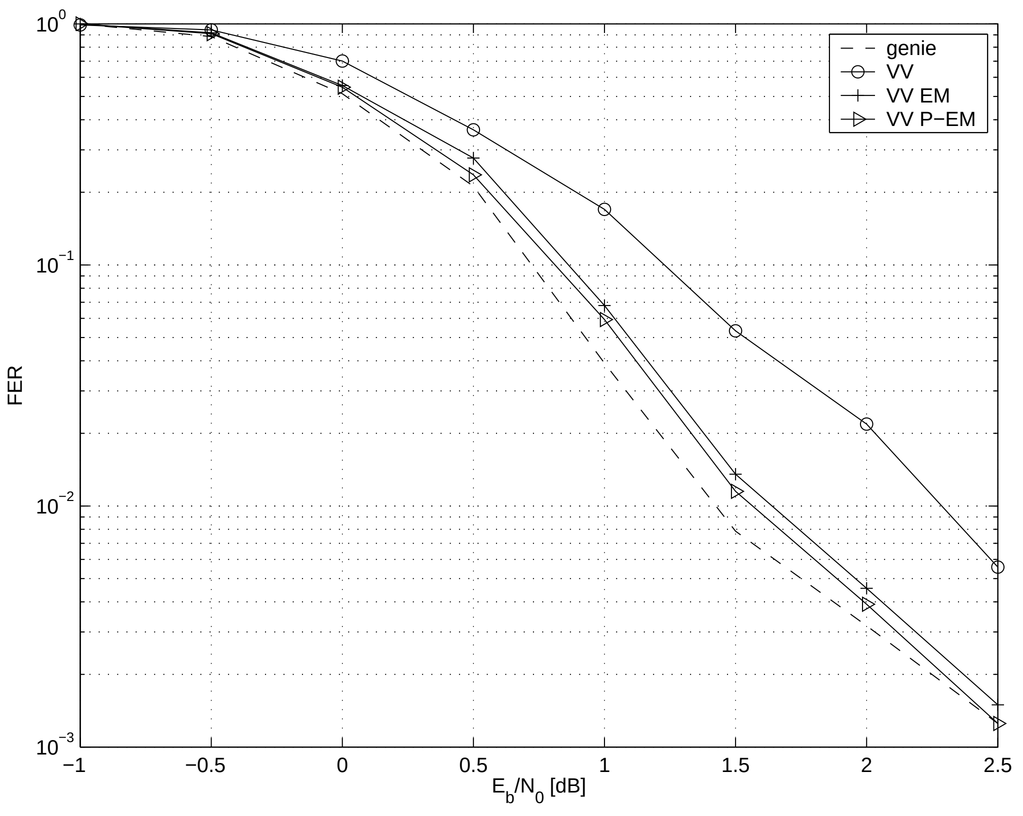


Figure 12.4: *Phase estimation: FER performance for fractional phase estimation after 10 EM iterations.*

- The Viterbi&Viterbi (VV) algorithm (see Chapter 9, section 9.2),
- The embedded EM algorithm (see Chapter 9, section 9.3), initialized by the VV algorithm,
- The parallel embedded EM algorithm (see Chapter 10), with two initial estimates: one from the VV algorithm and a second estimate obtained by rotating the VV estimate over $k\pi/4$, where $k \in \mathbb{Z}_0$ is selected such that the resulting estimate gives rise to an estimation error in the interval $[-\pi/4, +\pi/4]$.

The MSEE performance after 10 decoding iterations is shown in Fig. 12.2. The VV algorithm results in a MSEE that is far away from the MCRB. The EM algorithm is able to reduce the MSEE. A small extra gain is achieved by the application of the P-EM algorithm. However this latter approach doubles the computational cost. Note that the P-EM algorithm attains the MCRB for $E_b/N_0 > 2$ dB. Hence, beyond 2 dB, no other estimation algorithm can outperform the P-EM algorithm in terms of MSEE performance.

The corresponding BER is shown in Fig. 12.3: in terms of BER, the VV algorithm gives rise to a degradation of up to 1 dB. The EM algorithm is again able to reduce the degradation to around 0.5 dB, while the P-EM algorithm results in a BER degradation of less than 0.2 dB. So, even-though the computational complexity related to estimation is fairly large, from a BER point of view, the P-EM algorithm is required to reduce the degradations to an acceptable level.

Let us now see how this all translates in FER performance (see Fig. 12.4): again the VV algorithm leads to a degradation (now less than 1 dB), while both variations of the EM algorithm lead to roughly the same FER performance.

Phase ambiguity resolution

Of course, the above problem of fractional phase estimation is not realistic. In reality the phase ambiguity has not alway been resolved perfectly. Before we move on to more realistic scenarios, we first tackle the problem of phase ambiguity resolution, whereby we assume that $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$. Again τ is perfectly known to the receiver. We will consider three code-aided algorithms to perform phase ambiguity resolution:

- the discrete EM algorithm from Chapter 10, section 10.4.2;

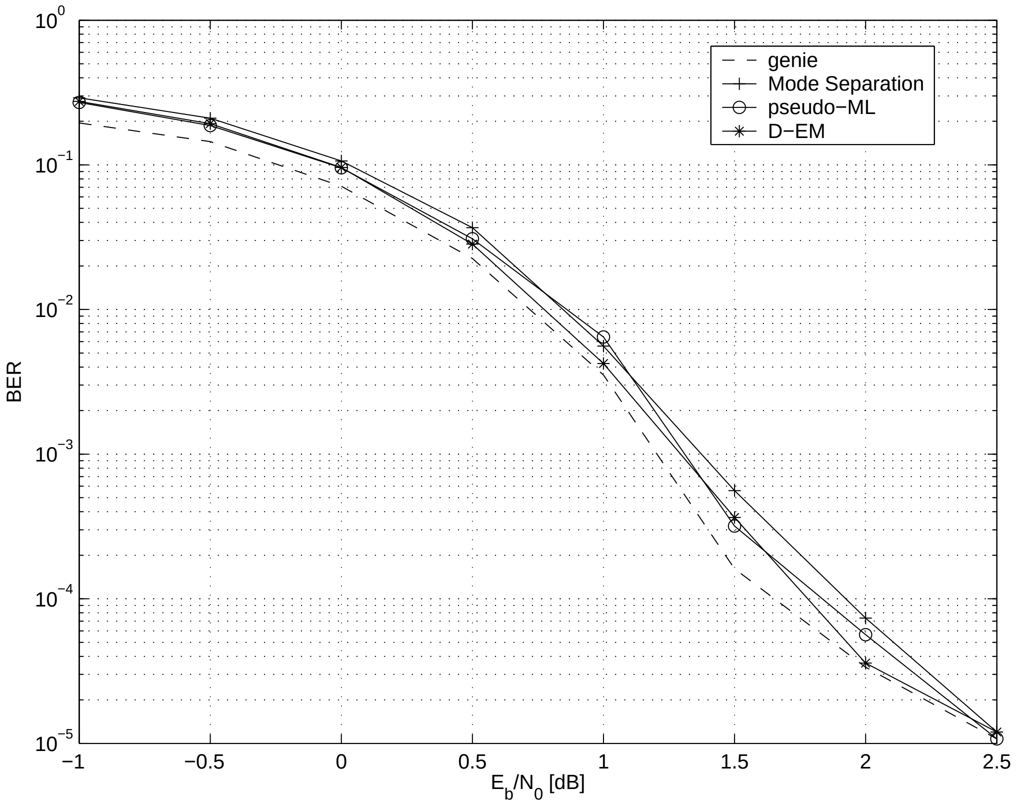


Figure 12.5: *Phase ambiguity resolution: BER performance after 10 decoding iterations.*

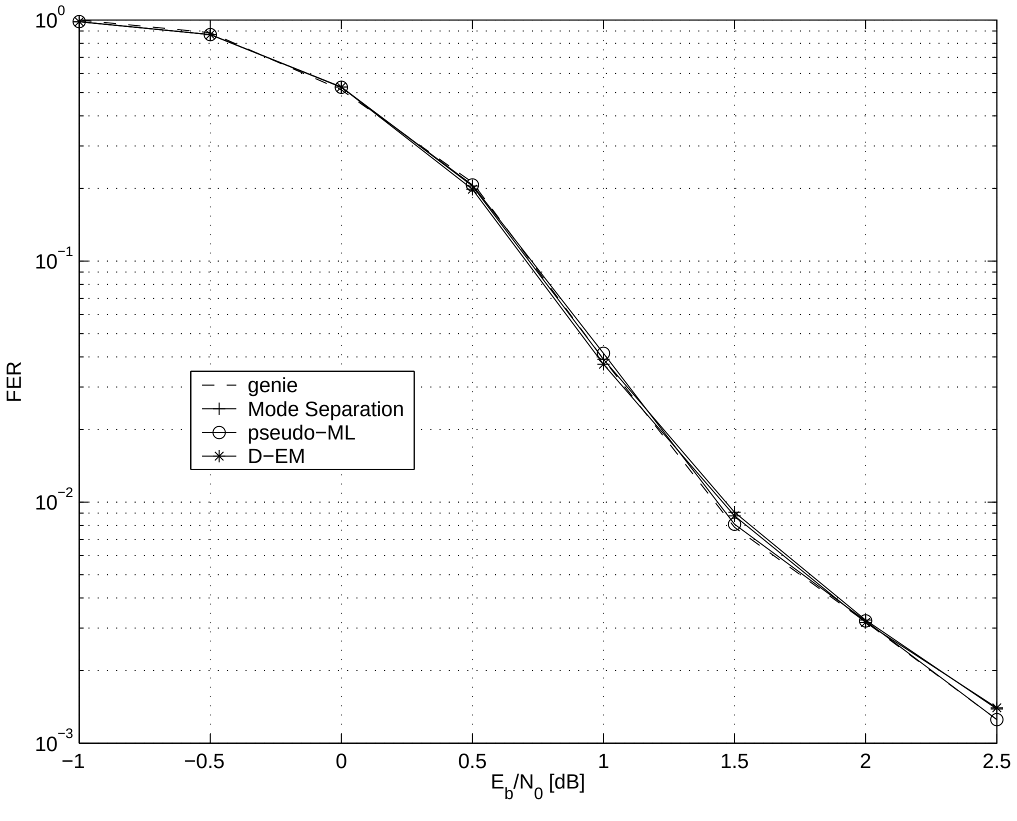


Figure 12.6: *Phase ambiguity resolution: FER performance after 10 decoding iterations.*

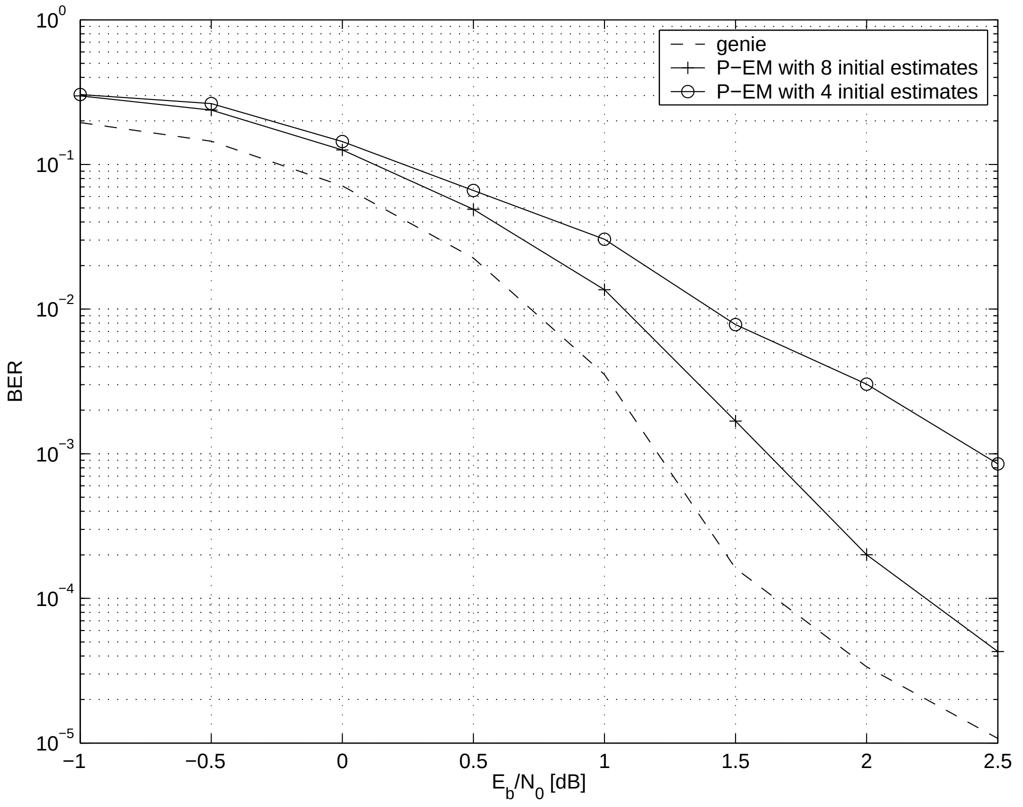


Figure 12.7: *Phase estimation: BER performance for total phase estimation after 10 EM iterations.*

- the Mode Separarion algorithm [117];
- the pseudo-ML algorithm from [127].

The latter two algorithms were described at the end of the previous chapter. These three algorithms operate in a similar way: for each of the four possible hypotheses, the packet is decoded using the synchronized factor graph. The outputs of the decoder are used to compute a certain metric. Based on this metric, a decision w.r.t. the correct hypothesis is made.

To reduce the computational complexity, we perform one decoding iteration for each of the four hypotheses, then make a decision w.r.t. the phase ambiguity and then perform the remaining nine iterations for the selected estimate of the ambiguity. This requires a total of $4 + 9 = 13$ decoding iterations, as opposed to 40 decoding iterations for the full-blown algorithms.

The BER (FER) performance of the three algorithms is shown in Fig. 12.5 (Fig. 12.6). The different algorithms lead to roughly the same performance. Observe the near-perfect FER performance: even without training symbols, the ambiguity does not cause a noticeable degradation.

Total phase estimation

We now combine the ideas from the previous paragraphs to perform phase estimation, without any assumptions regarding the possible values of the phase (or the estimation error). As we are not using pilot symbols, there is no conventional algorithm that is able to estimate the phase. Furthermore, the Mode Separation and pseudo-ML algorithms from the previous section cannot be applied[2], as they are hypothesis testing algorithms, not estimation algorithms. Hence, we can only apply the parallel EM algorithm: we first estimate the phase with the VV algorithm to obtain an estimate $\hat{\theta}_{VV}$ in the interval $[-\pi/4, +\pi/4]$. Then M_{EM} initial estimates are constructed as $\hat{\theta}_k^{(0)} = \hat{\theta}_{VV} + k2\pi/(M_{EM})$, $k = 0, \ldots, M_{EM} - 1$. For each of these estimates, we perform a single decoding iteration in the synchronized factor graph of the receiver. We then select the 'best' initial estimate according to the parallel EM algorithm, and perform the remaining 9 EM iterations using only that initial estimate. After each iteration, the phase estimate is updated.

[2]To be fair, we could replace the decision rule of the parallel EM algorithm with those of the Mode Separation or Pseudo-ML techniques. Considering the previous results from this section, we expect little performance difference as compared to the parallel EM algorithm.

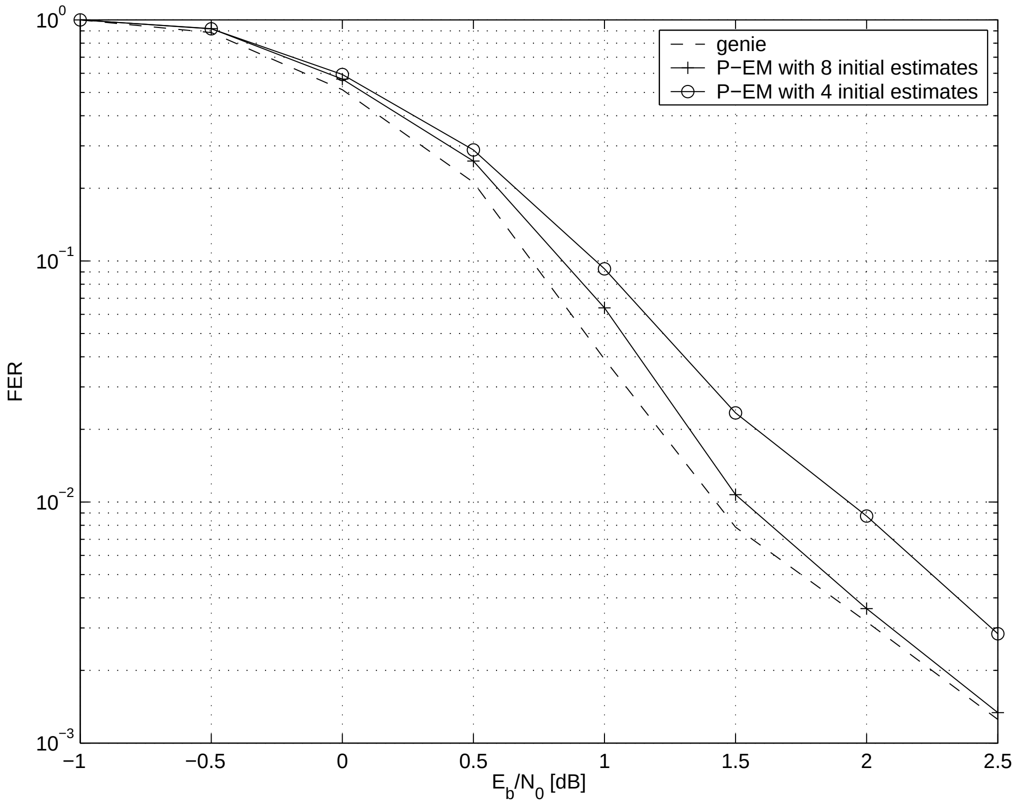


Figure 12.8: *Phase estimation: FER performance for total phase estimation after 10 EM iterations.*

In Fig. 12.7 and Fig. 12.8 BER and FER results are shown for $M_{EM} = 4$ (the smallest possible value) and $M_{EM} = 8$. It is clear that in order to achieve small FER degradations, only $M_{EM} = 8$ gives acceptable results. Note that we have not optimized M_{EM}. In practice, one would try to minimize $M_{EM} \geq 4$ to achieve a target FER.

12.3.4 Delay estimation

In this second phase, we assume the receiver has perfect knowledge of the carrier phase θ.

Fractional delay estimation

As a first step, we consider fractional delay estimation: assume the receiver has perfect knowledge of the carrier phase θ, and perfect frame synchronization has been achieved. Hence, the initial delay error is in the interval $[-T/2, +T/2]$. Two estimation algorithms will be considered:

- The Oerder&Meyr (OM) algorithm (see Chapter 9, section 9.2),
- The embedded EM algorithm (see Chapter 9, section 9.3), initialized by the OM algorithm. The maximization from (9.54) is performed by the well-known Newton-Raphson algorithm.

In parallel to the results for fractional phase estimation, we first show the MSEE performance in Fig. 12.9: the OM estimator is far away from the MCRB, while the EM estimator attains the MCRB for $E_b/N_0 > 1.5$ dB. Hence, beyond this SNR, no other estimation algorithm can outperform the EM algorithm in terms of MSEE performance.

The corresponding BER is shown in Fig. 12.10: in terms of BER, the OM algorithm gives rise to a degradation of up to 0.8 dB. The EM algorithm is again able to reduce the degradation to less than 0.1 dB for all considered SNR. FER performance is depicted in Fig. 12.11: the degradation of around 0.3 dB from OM estimation is almost completely removed by the EM algorithm.

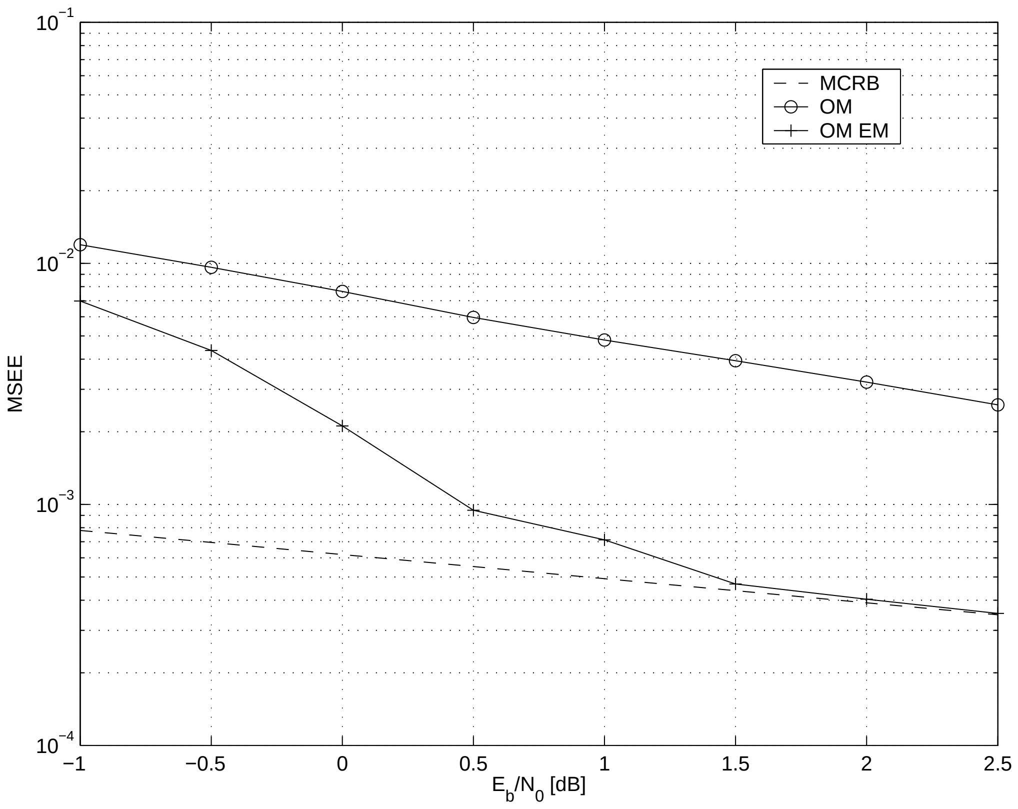


Figure 12.9: *Delay estimation: MSEE performance for fractional delay estimation after 10 EM iterations.*

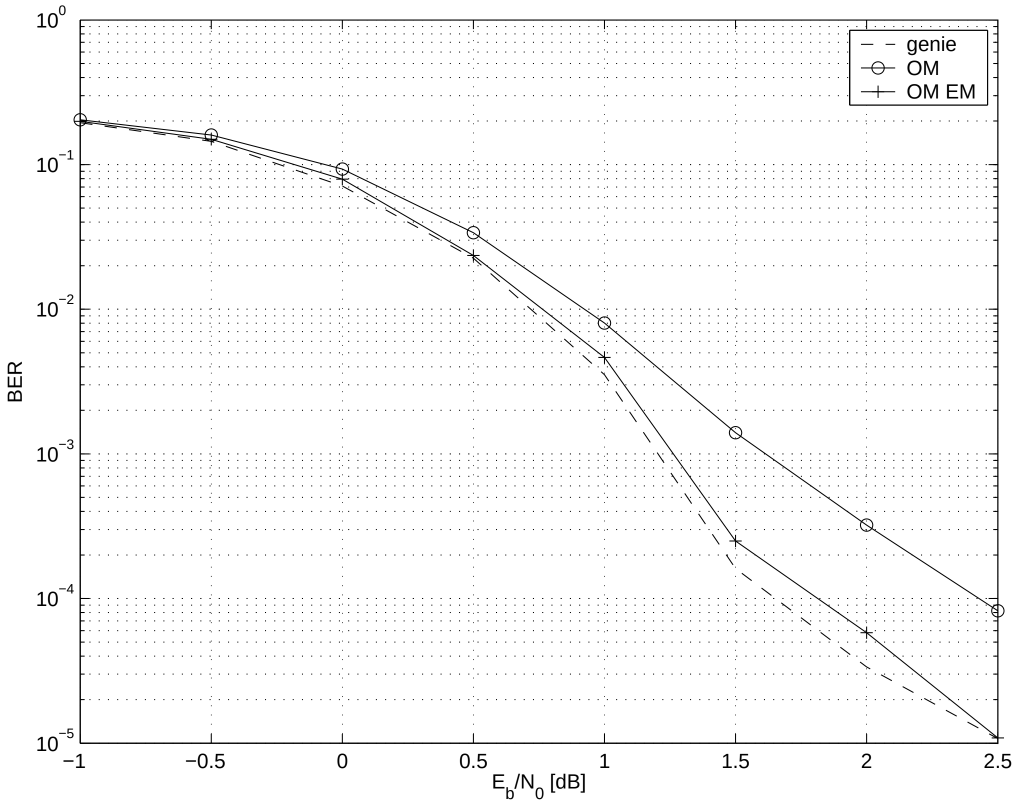


Figure 12.10: *Delay estimation: BER performance for fractional delay estimation after 10 EM iterations.*

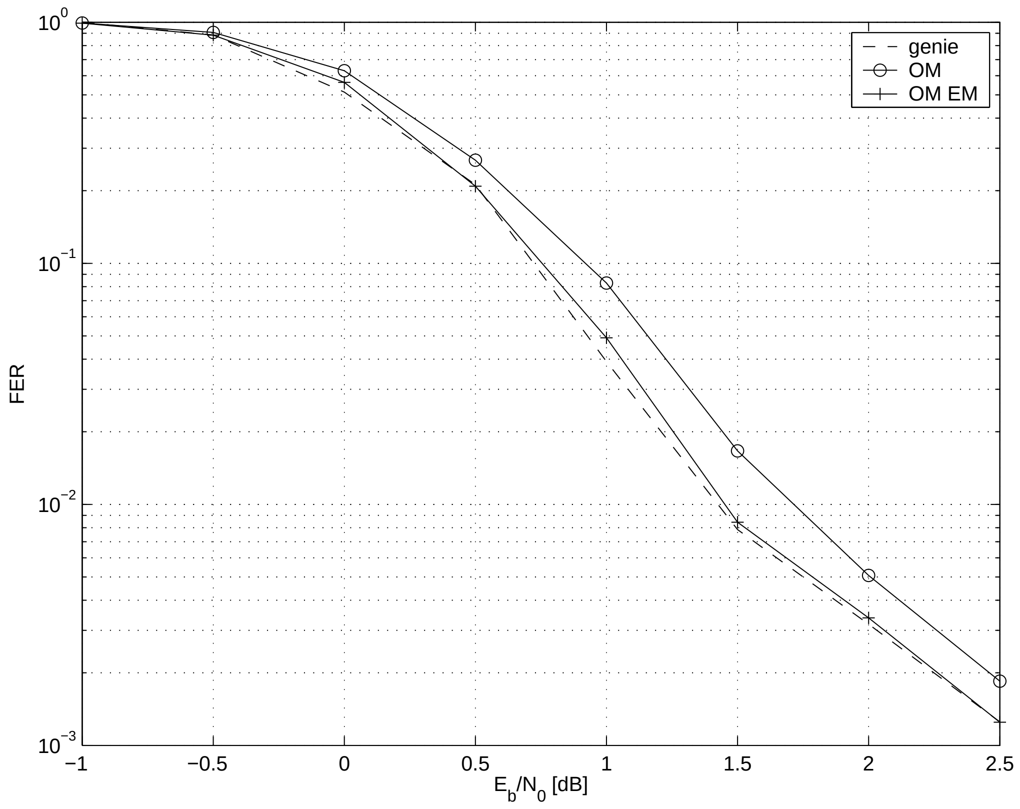


Figure 12.11: *Delay estimation: FER performance for fractional delay estimation after 10 EM iterations.*

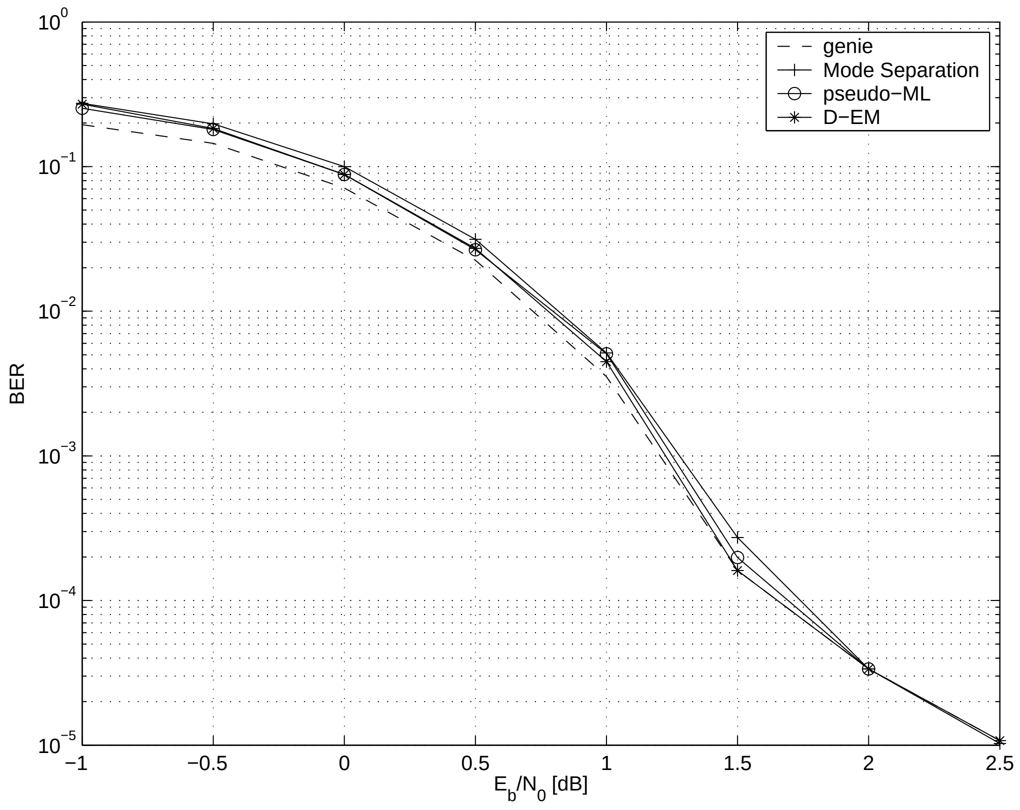


Figure 12.12: *Frame synchronization: BER performance after 10 decoding iterations.*

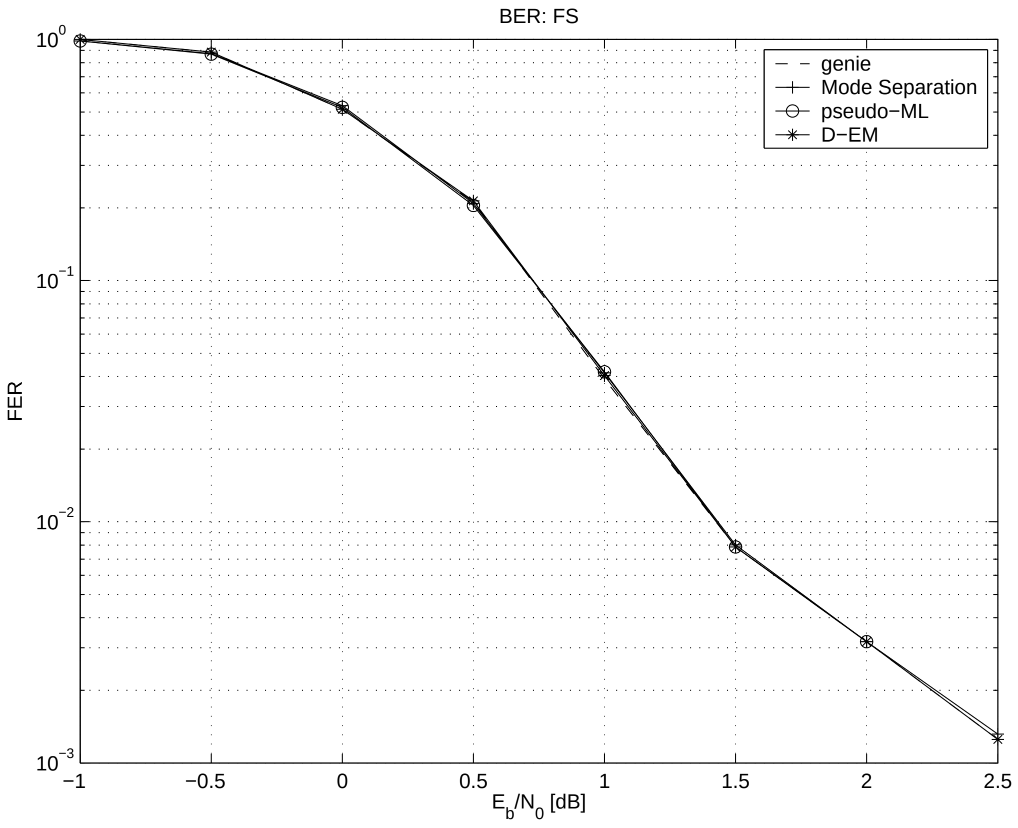


Figure 12.13: *Frame synchronization: FER performance after 10 decoding iterations.*

Frame synchronization

Again, the problem of fractional delay estimation is not realistic. In reality we cannot assume that perfect frame synchronization has been achieved. Before we move on to more realistic scenarios, we first tackle the problem of frame synchronization, whereby we assume that $\tau \in \{-2T, \dots, 2T\}$. We will consider three code-aided algorithms to perform frame synchronization: the discrete EM algorithm as well as the two algorithms from recent technical literature: Mode Separation and pseudo-ML (as described at the end of the previous chapter).

To reduce the computational complexity, we perform one decoding iteration for each of the five hypotheses, then make a decision w.r.t. the frame position and then perform the remaining nine iterations for the selected estimate of the frame position. This requires a total of $5 + 9 = 14$ decoding iteration, as opposed to 50 decoding iterations for the full-blown algorithms.

The BER (and FER) performance of the three algorithms is shown in Fig. 12.12 (and Fig. 12.13). The different algorithms lead to roughly the same performance. Observe the near-perfect FER performance: even without training symbols, frame synchronization can be performed without a noticeable degradation.

Total Delay estimation

Combining the results from the previous sections, we construct the following code-aided algorithm to perform joint delay estimation and frame synchronization: we estimate the fractional part of τ through the OM algorithm, resulting in $\hat{\tau}_{OM}$. Then M_{EM} initial estimates can be constructed as $\hat{\tau}_k^{(0)} = \hat{\tau}_{OM} + kT/\tilde{M}$ for suitable values of k and $\tilde{M}$, so that all $\hat{\tau}_k^{(0)}$ are in the interval $[-M_1 T, \dots, M_2 T]$. For each of these estimates, we decode the packet. We then select the best initial estimate according to the parallel EM algorithm, and perform the remaining 9 EM iterations using only that initial estimate. After each iteration, the delay estimate is updated.

In Fig. 12.14 and Fig. 12.15 results are shown for $M_{EM} = 5$ and $\tilde{M} = 1$, which are the minimal values to perform joint delay estimation and frame synchronization (as demonstrated in [110]). A non-negligible BER degradation is noticeable. This degradation can be removed by making a decision w.r.t. the best initial estimate after more than one decoding iteration, at an increase in computational complexity. On the other hand, the FER performance is quite acceptable.

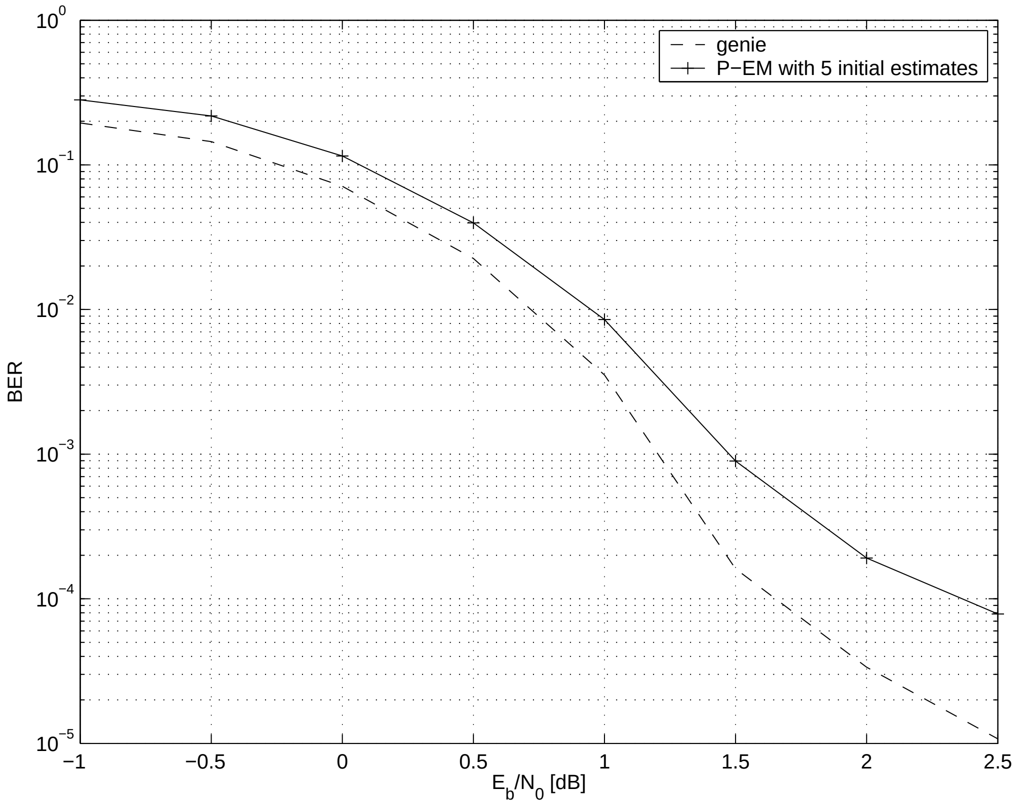


Figure 12.14: *Delay estimation: BER performance for total delay estimation after 10 EM iterations.*

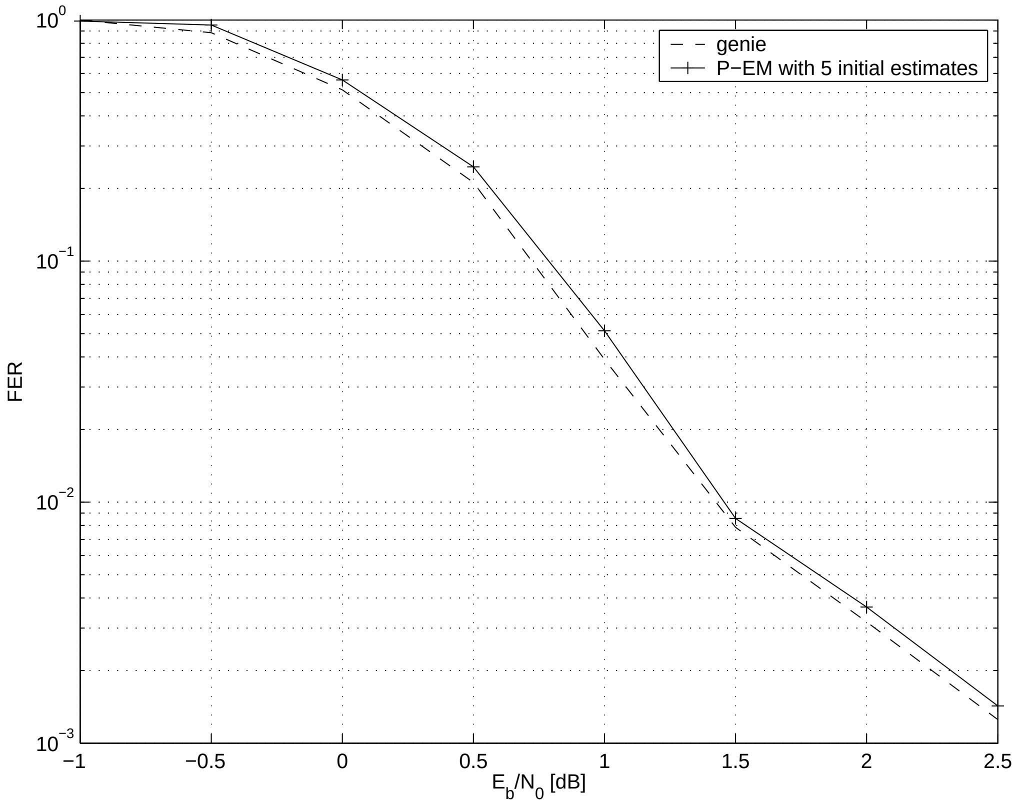


Figure 12.15: *Delay estimation: FER performance for total delay estimation after 10 EM iterations.*

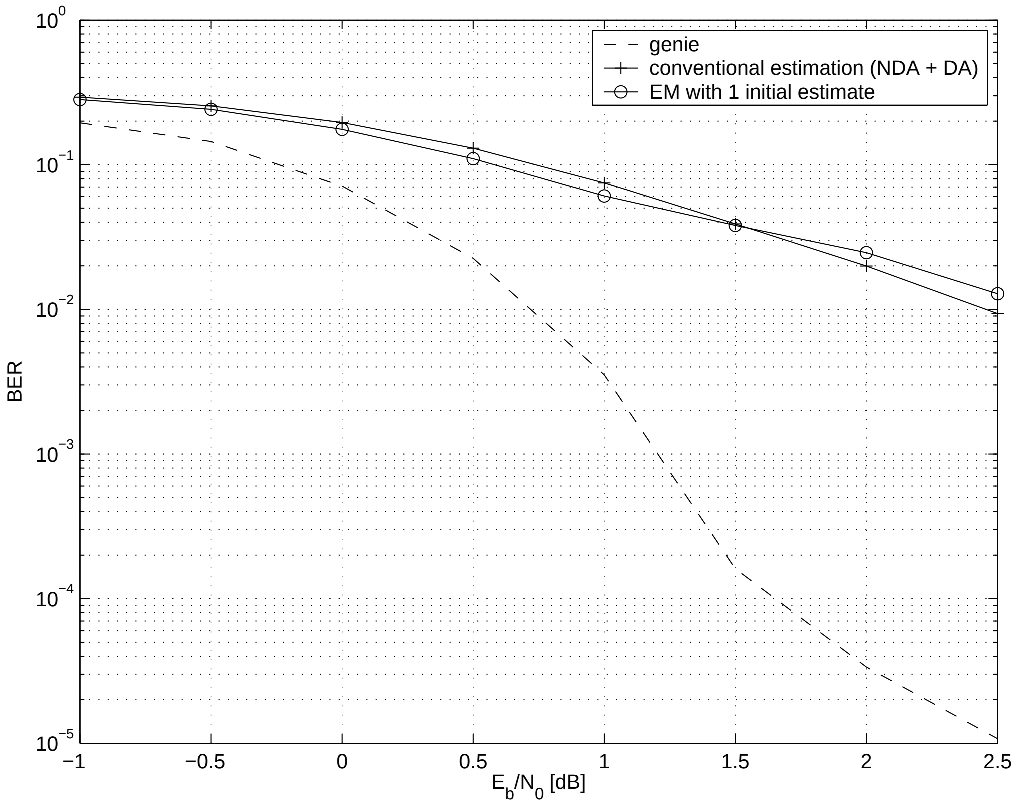


Figure 12.16: *Joint delay and phase estimation: BER performance after 10 EM iterations for conventional estimation and EM estimation. A pilot sequence of 10 pilot symbols was assumed.*

12.3.5 Joint delay and phase estimation

As a final step, we will consider joint delay and phase estimation, including frame synchronization and phase ambiguity resolution. Clearly, when no pilot symbols are present, conventional estimation algorithms cannot be applied. So, let us first consider a situation where we include 10 pilot symbols in the data stream. This will result in a rate/power loss of around 0.12 dB. Performing estimation without exploiting code properties leads to BER (resp. FER) results shown in Fig. 12.16 (resp. 12.17) (denoted 'conventional estimation'). We see very high degradations, both in terms of BER and FER. Suppose we now take the obtained estimate from the conventional estimation algorithms as an initial estimate for our code-aided EM estimator (denoted 'EM with 1 initial estimate') in Fig, 12.16-12.17. The EM estimator was not able to reduce the degradations because of the poor quality of the initial estimates.

Let us now see what happens when we perform the parallel EM algorithm *without* using pilot symbols. We consider the same number of initial estimates as for total delay estimation and total phase estimation: 5 initial estimates for τ, and 8 initial estimates for θ. We make a decision w.r.t. initial estimate after the n-th decoding iteration, and show results for $n = 1, \ldots, 10$.

In Fig. 12.18 and Fig. 12.19 we show BER and FER performance results, respectively. Different curves correspond to different iterations when the decision w.r.t. the frame position and phase ambiguity is made (i.e., after 1 iteration in the top-most curve, after 2 iterations in the curve below and so forth). The topmost curve leads to the largest degradations, but at the same time corresponds to the lowest complexity.

While the degradations compared to the genie receiver are non-negligible, the reader should be aware of the fact that no pilot symbols are used in any of these simulations. The code is able to synchronizes itself. In that respect, the performance of the code-aided estimation algorithms is very impressive.

12.4 Channel estimation: frequency-selective channel

12.4.1 System set-up

As our research in this area was mainly focused on DS/SS systems, the system model will be slightly different. We will consider a convolutional code which is recursive and systematic with octal generators $(23, 35)_8$ and constraint

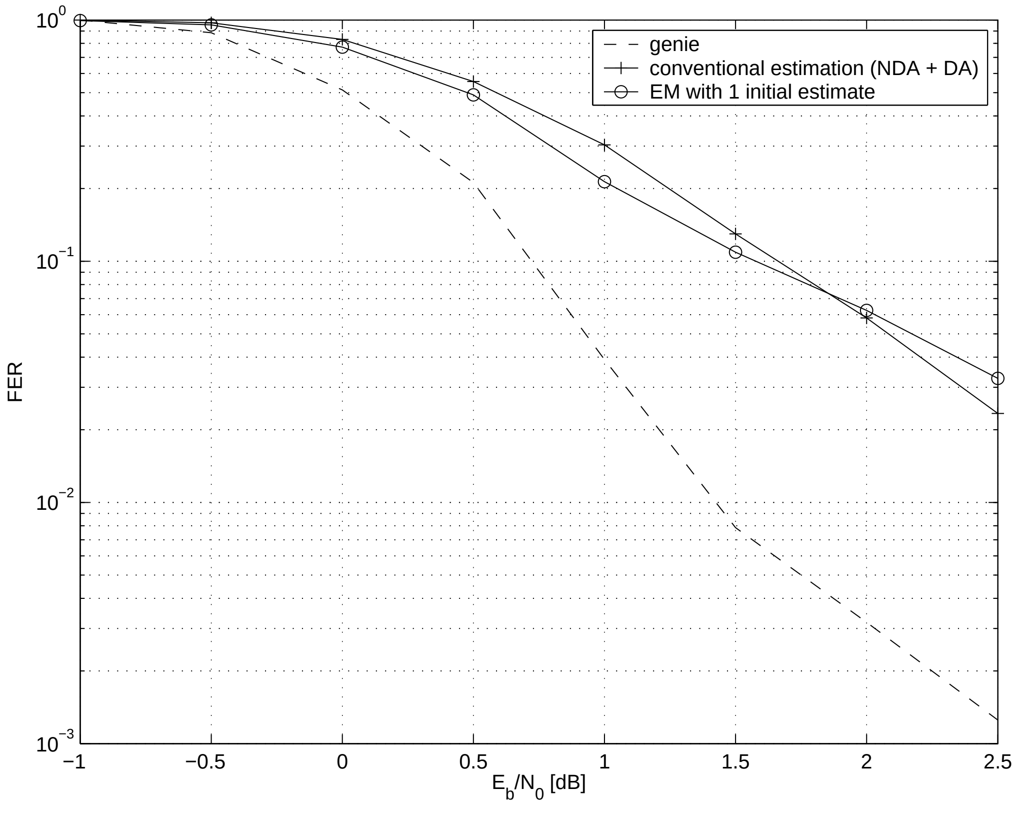


Figure 12.17: *Joint delay and phase estimation: FER performance after 10 EM iterations for conventional estimation and EM estimation. A pilot sequence of 10 pilot symbols was assumed.*

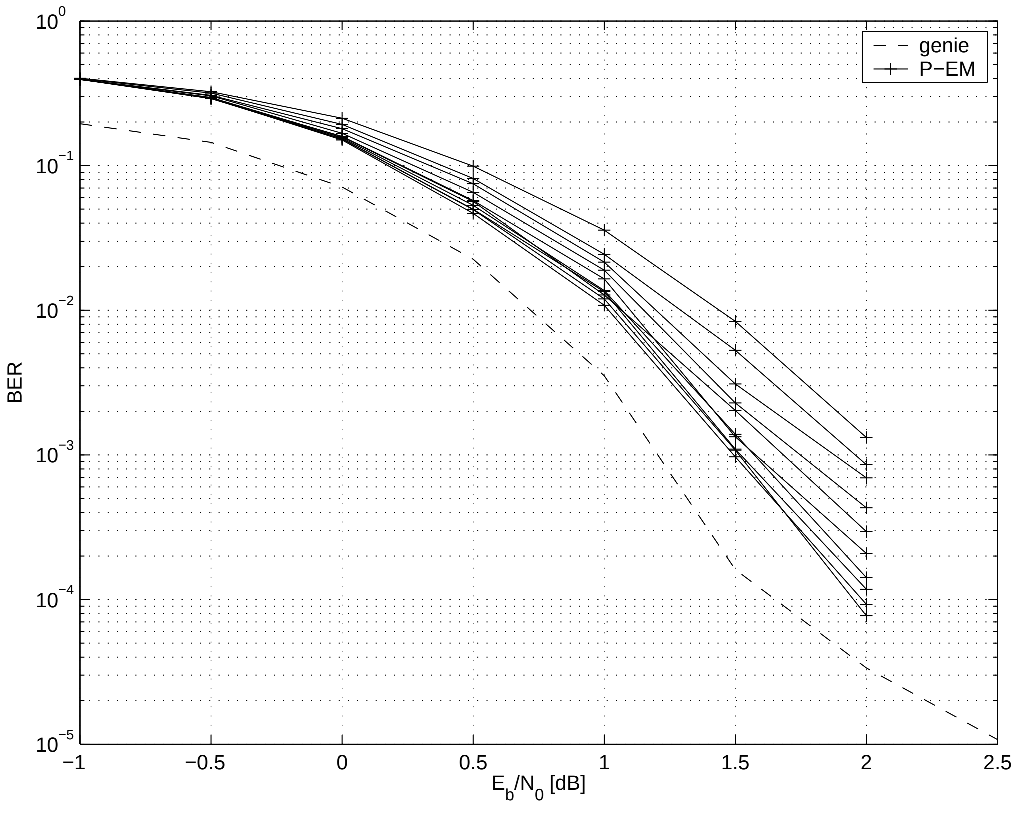


Figure 12.18: *Joint delay and phase estimation: BER performance after 10 EM iterations. There are no pilot symbols present.*

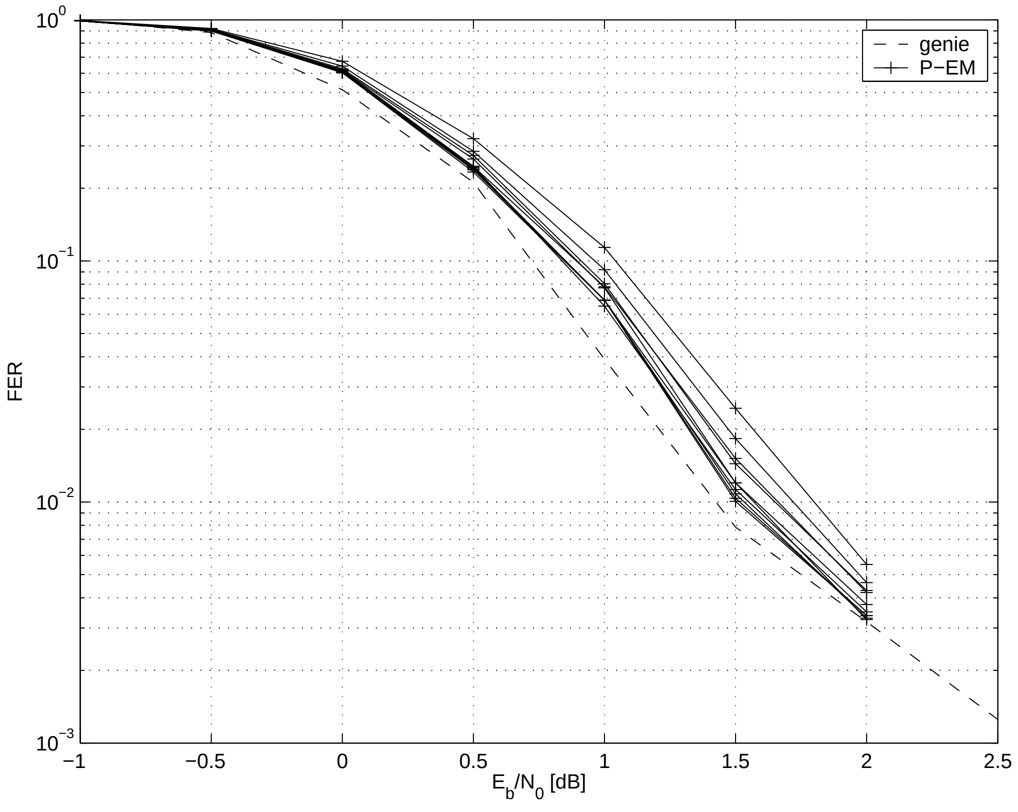


Figure 12.19: *Joint delay and phase estimation: FER performance after 10 EM iterations. There are no pilot symbols present.*

length $\nu = 5$. The information $N_b = 180$ bits are encoded with this rate $1/2$ code, yielding $N_c = 360$ coded bits. These bits are interleaved prior to Gray-mapping onto an 8-PSK constellation. The 120 8-PSK symbols are preceded by 10 pilot symbols and are placed on a unit-energy pulse. We consider a pulse of the form

$$p(t) = \frac{1}{N_g} \sum_{k=0}^{N_g} \rho_k p_c (t - kT/N_g) \tag{12.2}$$

where $p_c(t)$ is a square-root cosine roll-off pulse with roll-off factor 0.3 (30%) and we set $N_g = 7$ with $[\rho_0, \ldots, \rho_6] = [-1, +1, -1, +1, +1, +1, -1]$. The resulting signal is transmitted over an equivalent complex baseband channel

$$h_{BB}(t) = \sum_{l=0}^{L-1} \alpha_l \delta (t - \tau_l). \tag{12.3}$$

As different channels lead to different BER and FER performance, we have chosen to select a fixed instantiation of the channel with $L = 3$ taps with

$$[\alpha_0, \alpha_1, \alpha_2] = [-0.299 - j0.567, 0.307 + j0.609, 0.301 + j0.378] \tag{12.4}$$

and

$$[\tau_0, \tau_1, \tau_2] = [1.5, 3.5, 6] * T/N_g. \tag{12.5}$$

The detector consists of an augmented MMSE equalizer (as described in Chapter 5), and a sum-product demodulator/decoder.

Estimator set-up

To estimate the channel impulse response, we will consider a structured estimator, based on the SAGE algorithm, that estimates the channel gains and propagation delays. The initial estimate is purely data-aided (see Chapter 9, section 9.2.1.2): for the initial estimate, the maximization w.r.t. the delays is performed by a line search in the range[3]

[3]Hence, strictly speaking, we will not consider frame synchronization.

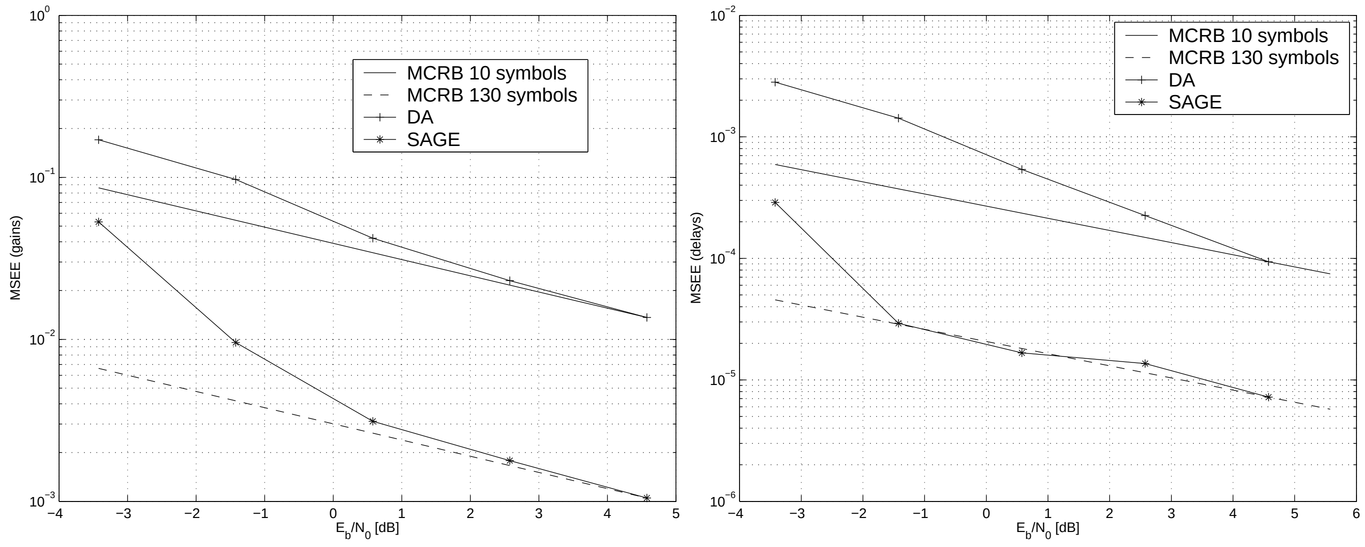


Figure 12.20: *Channel parameter estimation - MSEE performance for gain estimation (left) and delay estimation (right).*

$\tau_k \in [0, T]$. Subsequent maximizations w.r.t. the propagation delays are performed through the Newton-Raphson algorithm. For frequency-selective channels, we have to be less ambitious with respect to the number of training symbols, as compared to frequency-flat channels: too few pilot symbols will result in unreliable initial estimates, while too many pilot symbols will degrade the system performance because of the reduction of power and bandwidth efficiency[4]. For our system model, we have found that 10 pilot symbols is a reasonable trade-off: the resulting loss amounts to around 0.35 dB. We will perform 6 decoding iterations. After each decoding iteration, we update all the channel parameters according to the embedded SAGE algorithm (see Chapter 10).

12.4.2 MSEE performance

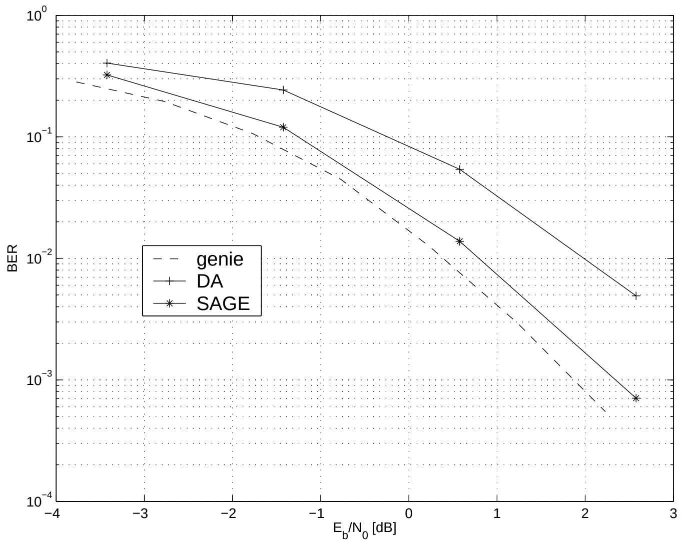
Let us start with MSEE performance. In Fig. 12.20, we show the MSEE of the estimated channel gains (on the left) and propagation delays (on the right). We have added the MCRBs corresponding to 10 symbols (the pilot sequence) and to 130 symbols (the entire frame). We see that the DA estimates result in an MSEE that is above the 10-symbols MCRB, and converges to that MCRB for increasing SNR, as we would expect. When we apply the SAGE estimator, we come closer to the 130-symbols MCRB. We see that at high SNR, we achieve optimal MSEE performance, both for delay and for gain estimation.

12.4.3 Error rate performance

We will now see how the impressive MSEE results translate into BER and FER performance. We remind that, due to pilot insertion, we expect a performance loss (also: rate loss, pilot insertion loss) of at least 0.35 dB as compared to a genie receiver (i.e., a receiver that knows the channel perfectly and thus requires no pilot symbols). In Fig. 12.21, we plot BER and FER performance for three scenarios: the genie receiver, the receiver where the channel parameters are estimated exploiting only the pilot symbols (DA estimator), and the SAGE estimator. We see that the DA estimator leads to degradations of around 1 dB, both in terms of BER and FER performance. The SAGE estimator is able to reduce this degradation to around the rate loss due to pilot insertion.

We conclude that the SAGE estimator can achieve good performance in terms of MSEE, BER and FER, but at a cost of bandwidth and power efficiency. In contrast to the frequency-flat channel, we were not able to create a SAGE estimator that does not require pilot symbols. However, should a reliable channel estimation algorithm be found that requires less (or no) pilot symbols, it can be used in conjunction with our code-aided SAGE estimator.

[4]Assuming the total power to be allocated to the packet to be constant.



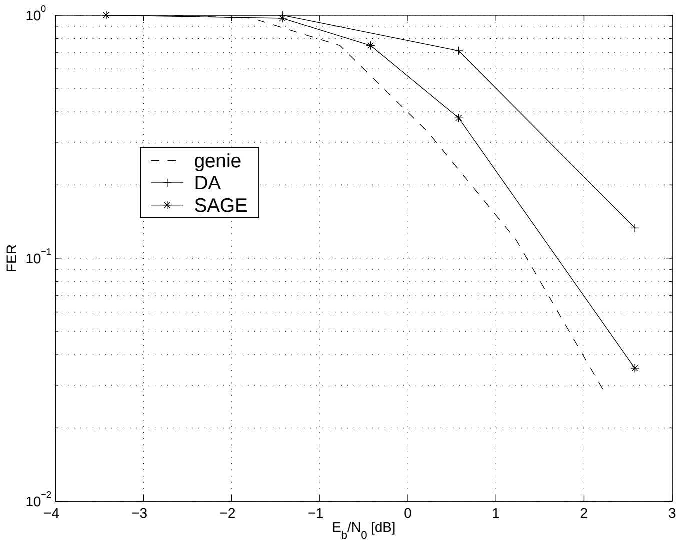


Figure 12.21: *Channel parameter estimation - BER performance (left) and FER performance (right)*

12.5 Rate detection

12.5.1 System Model

The last problem we will consider is that of rate detection (see Chapter 11, section 11.2). The transmitter sends a sequence of N_s symbols using a transmit pulse $p_T(t)$, corresponding to symbol rate $1/T$. The symbol interval T belongs to a finite set of equiprobable values: $T \in S_T = \{T_{min}, \ldots, T_{max}\}$. The receiver knows the set S_T .

The channel is assumed to be frequency-flat. We consider a fully digital receiver whereby the signal $r_{BB}(t)$ is band-limited through analog filtering and sampled at a fixed rate $1/T_s$:

$$r_{BB}(kT_s) = Ae^{j\theta}s_T(kT_s - \tau) + n(kT_s) \tag{12.6}$$

with $E[n(kT_s)n^*(lT_s)] = N_0T_s\delta_{k-l}$ and $s_T(t)$ the transmitted signal, corresponding to an unknown symbol rate $1/T$. For convenience, issues related to frame synchronization and phase ambiguity resolution will not be addressed in this section.

12.5.2 Rate detection algorithms

We will consider the three rate detection (RD) algorithms that were described in Chapter 11, section 11.2:

- A cyclic correlation-based algorithm from [121] performs rate detection in front of the matched filter bank. This algorithm requires no knowledge of τ, A or θ.
- A second algorithm is based on a low-SNR approximation of the likelihood function [124]. This algorithm requires no knowledge of τ, A or θ. It is computationally more complex than the cyclic correlation approach from [121], since now the incoming signal has to be filtered by each of the matched filters.
- A last algorithm is the discrete EM algorithm from Chapter 11, section 11.2. The received signal is applied to the matched filter bank, and to each filter output, an algorithm for estimating τ, A and θ is applied. For reasons of computational complexity, we treat the data as uncoded during rate detection.

The performance measure we consider is the RD error probability (RDEP), i.e., the fraction of frames for which the symbol rate is not correctly detected. As a detection error results in the loss of an entire frame, the RDEP should be sufficiently low. More specifically, if we denote the BER of a (coded or uncoded) system under perfect symbol rate detection by BER_0, then the BER in the presence of occasional RD errors is upper-bounded by

$$\begin{aligned} BER_{RD} &< BER_0(1 - RDEP) + 1 \times RDEP & (12.7) \\ &\approx BER_0\left(1 + \frac{RDEP}{BER_0}\right). & (12.8) \end{aligned}$$

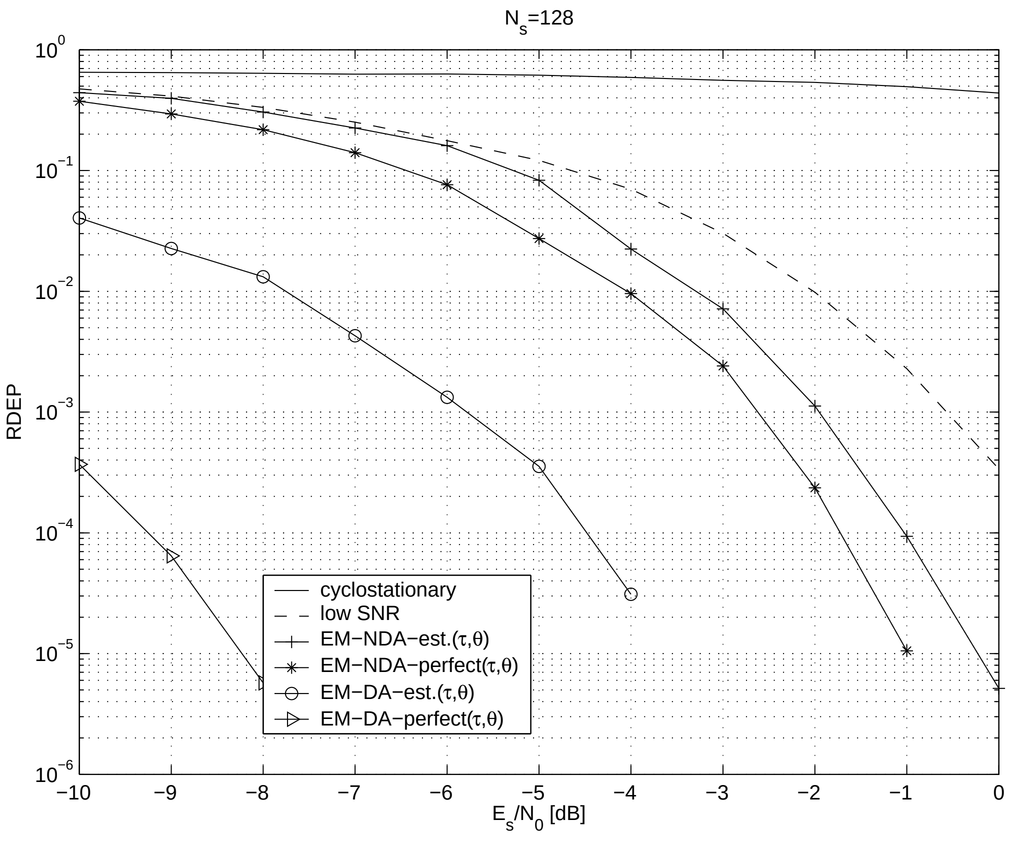


Figure 12.22: *Rate detection error probability (RDEP) as a function of the SNR for block length* $N_s = 128$

Consequently, in order to obtain a low BER degradation due to RD, the ratio $RDEP/BER_0$ should be below 1, in which case we obtain $BER_{RD} < 2BER_0$. A similar reasoning can be applied to frame error rate performance.

12.5.3 Performance results

We have carried out computer simulations for BPSK transmission, with $p_T(t)$ a square-root cosine roll-off pulse with roll-off $\alpha = 0.5$. We set $S = \{4T_s, 8T_s, 16T_s\}$ and consider frame lengths N_s= 128 and $N_s = 512$. The propagation delay τ and the carrier phase θ will be estimated using the conventional Oerder&Meyr [90] and Viterbi&Viterbi [92] algorithm, respectively. We assume $A = 1$ and A is known to the receiver.

In the discrete EM algorithm we discern four sub-cases: the propagation delay τ and the carrier phase θ are either known at the receiver or are estimated using conventional algorithms from [90] and [92], respectively. These cases are denoted by 'perfect(τ, θ)' and 'est(τ, θ)' in Fig. 12.22-12.23, respectively. On the other hand, the data symbols $\mathbf{a}$ are either known at the receiver (to be denoted 'EM-DA', for Data-Aided) or unknown (to be denoted 'EM-NDA', for Non-DA). Simulations for a given RD algorithm were halted after at least 100 rate detection errors.

Fig. 12.22 (resp. Fig. 12.23) displays RDEP for the different configurations for $N_s = 128$ (resp. $N_s = 512$). We observe that the cyclic correlation-based approach exhibits very poor performance in the considered SNR range. This is due to two factors: for low rates, excess noise will degrade the performance, while for high rates, the number of samples per symbol interval is too low to accurately detect T. The EM-NDA algorithm outperforms the low-SNR algorithm by about one order of magnitude. The performance of the NDA EM-based algorithm can be further improved by around 1 dB by applying more advanced estimation algorithms for τ and θ (see EM-NDA-perfect(τ, θ)). On the other hand, an improvement of around 3.5 dB is visible when the data symbols are assumed to be known at the receiver. This gives us some idea of the performance improvements attainable by exploiting code properties during RD. Combining these two (i.e., exploiting the code during RD and using superior algorithms for estimation), can give us a total improvement of around 7 to 8 dB. We remind that exploiting code properties during rate estimation gives rise to a significant computational overhead, so that the resulting computational complexity may not justify such a code-aided approach.

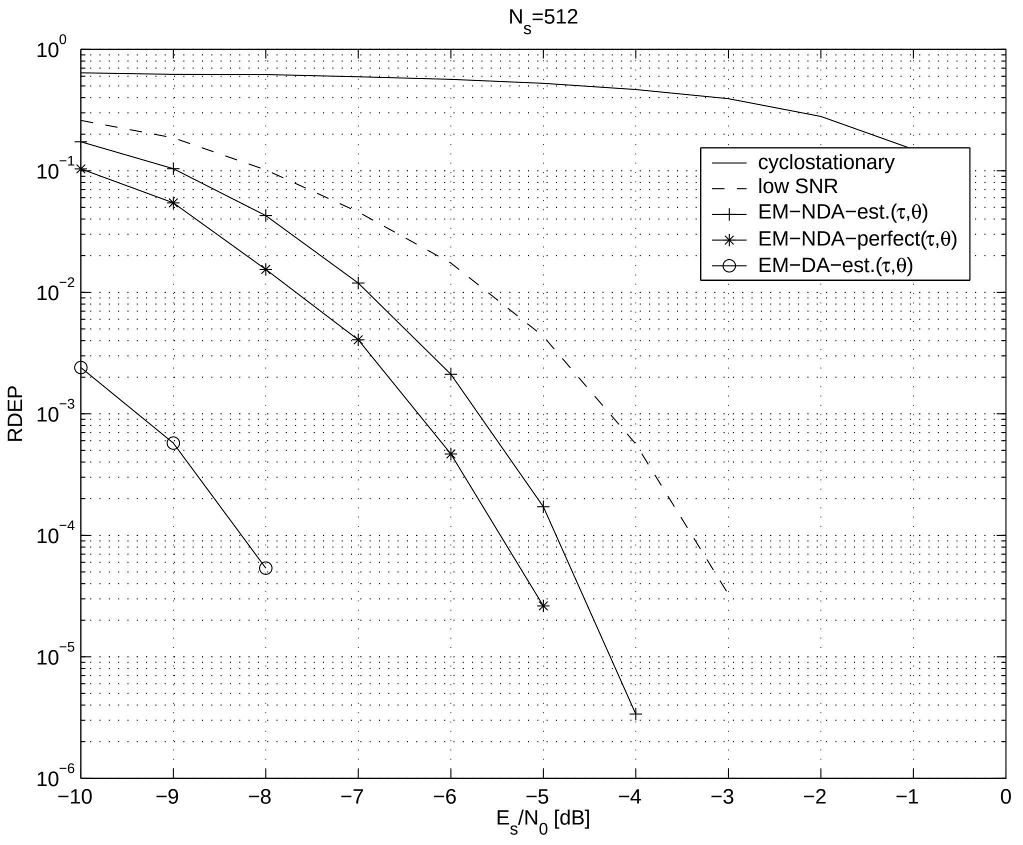


Figure 12.23: *Rate detection error probability (RDEP) as a function of the SNR for block length*$N_s = 512$

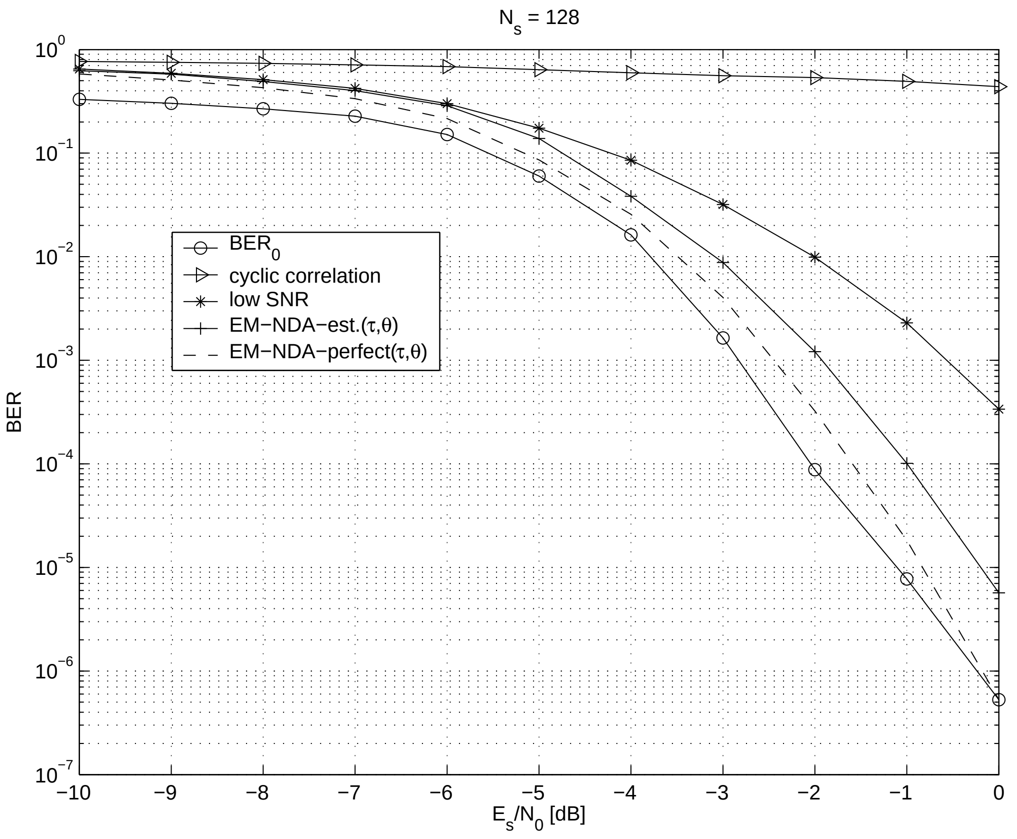


Figure 12.24: *BER performance for turbo code for* $N_s = 128$

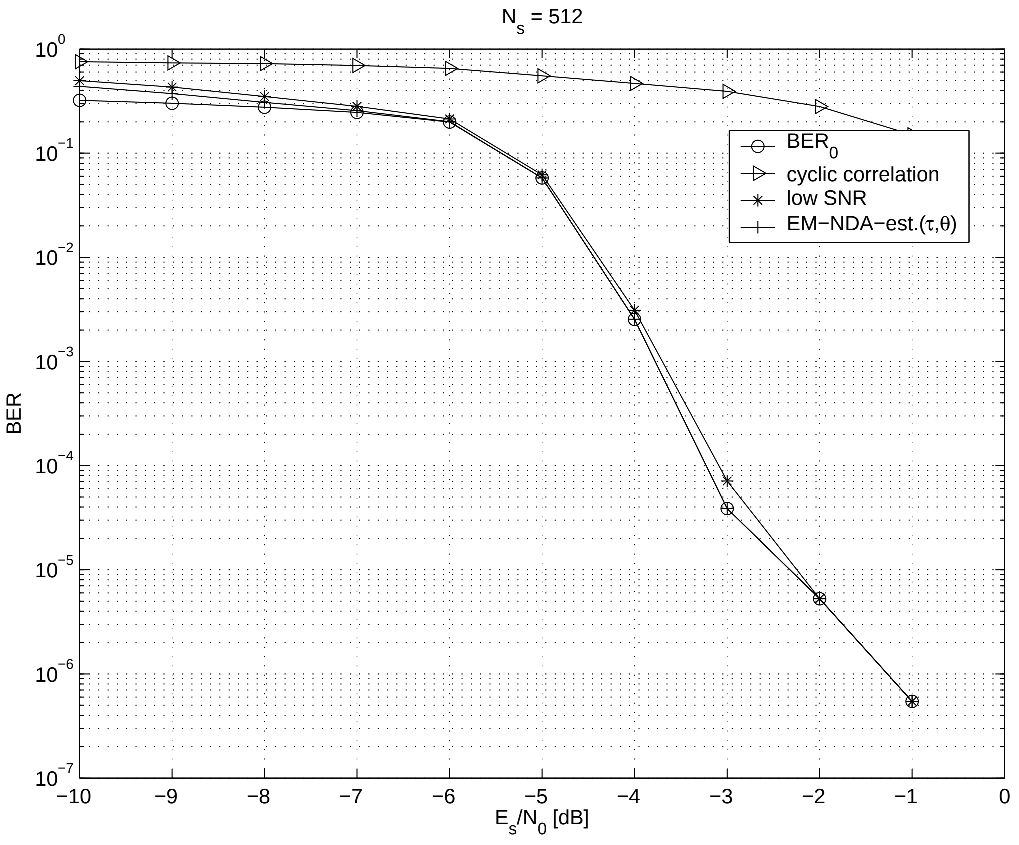


Figure 12.25: *BER performance for turbo code for* $N_s = 128$

In Fig. 12.24 (resp. Fig. 12.25), the impact of rate detection on the overall BER for a turbo-coded system[5] can be observed for $N_s = 128$ (resp. $N_s = 512$). For $N_s = 512$, both the low-SNR and EM-based algorithm yield excellent performance. For $N_s = 128$, the situation is somewhat different: the low-SNR method gives rise to a BER degradation of around 2 dB. Application of the EM-based algorithm reduces this degradation to around 1 dB. For the sake of illustration, we have included the BER performance when τ and θ are perfectly known at the receiver. The resulting degradation is around 0.5 dB. This means that by applying superior propagation delay and carrier phase estimation algorithms, a small reduction in BER degradation is possible. In order to reduce the degradation even further, one must consider code-aided rate detection algorithms.

12.6 Main Points

In this chapter, we have applied the code-aided EM and SAGE algorithms to some specific estimation problems. We first treated estimation of carrier phase and propagation delay for frequency-flat channels, where we illustrated that different estimation problems call for different estimation algorithms. When ample computational power is available, code-aided estimation can be performed, removing the need for training sequences in state-of-the-art error-correcting codes. This remarkable feat is mainly due to the inherent randomness of these codes.

For frequency-selective channels, a short pilot sequence is required to obtain an initial estimate of the channel parameters, resulting in a performance penalty. For the system we considered, any degradation due to unreliable channel estimation has been removed by the application of the SAGE estimator.

Finally, we investigated different rate detection algorithms. The conventional algorithms are all outperformed by the discrete EM algorithm, at the cost of some computational overhead. For some situations (e.g., bursts with many symbols), conventional techniques perform satisfactory.

While we have selected some examples that really make code-aided estimation shine, we stress that each problem requires a specially tailored estimation algorithm. This algorithm will be a trade-off between performance, affordable complexity and processing delay. The main message is that there is no hard and fast solution for all estimation problems.

[5]We consider a rate $1/3$ turbo code whereby the constituent convolutional codes are recursive, systematic and separated by a pseudo-random interleaver. The codes have generator polynomials $(21, 37)_8$ and constraint length 5. Only the first convolutional code is terminated.

Part IV

Concluding Remarks

Chapter 13

Open Issues and Loose Ends

13.1 Introduction

In this dissertation, we have tackled problems related to mode adaptation and estimation for coded multi-mode receivers. Although the algorithms we have derived are quite general, we have limited ourselves in many ways. For instance, we only considered a scenario with a single user, where transmitter and receiver are equipped with a single antenna. How will the detection and estimation algorithms behave in a multi-antenna and/or multi-user setting? We focused on linear modulations for a single carrier system. What modifications are required for multi-carrier systems? Also, the channel model was limited to a quasi-static block fading channel, which may be unrealistic in practical applications (e.g., when transmitter and/or receiver are moving). When the channel parameters and synchronization parameters are time-varying, other estimation techniques may be called for. Furthermore, we selected a particular set of parameters to estimate. Some parameters, such as oscillator frequency offsets, Doppler shifts, thermal noise variance were assumed to be known at the receiver. Additionally, although we have discussed mode adaptation (i.e., *how* the receiver can adapt to different transmit modes), we did not pay much attention on link adaptation (i.e., *when*, according to which criteria, to select a given mode). How much can be gained by clever link adaptation? How does all this translate into information-theoretical concepts? How much do we lose, in terms of capacity, by not having perfect knowledge of the channel parameters, synchronization parameters and/or transmit mode? This is related to the impact of imperfect channel knowledge at the transmitter: link adaptation requires some type of channel state information to be fed back from the receiver to the transmitter.

By its very nature, this dissertation cannot encompass all these aspects. In the current chapter, we briefly touch upon some of these issues.

13.2 Parameter estimation

13.2.1 Static parameters

The proposed algorithms can be easily altered/extended to suit different scenarios. Factor graph descriptions of almost any conceivable receiver configuration can be created. Once an observation model is constructed, the EM and SAGE algorithms can be obtained in a straightforward way. For instance, we have applied the SAGE algorithm to channel estimation and synchronization for multi-user, multi-antenna systems in [131–133]. The mathematical manipulations become somewhat cumbersome, but the principles remain the same: an iterative detector computes (approximations of) the APPs of the coded symbols. These are used in the evaluation of the E-step of the EM or SAGE algorithms.

Estimation of additional parameters such as thermal noise variance and frequency offsets can easily be included in the EM and SAGE estimation algorithms. Code-aided frequency offset estimation has already been described in [19] and applied in [134]. EM-based noise variance estimation has been addressed in [135].

13.2.2 Time-varying parameters

Code-aided estimation of time-varying parameters is a very challenging subject. A simple, yet illustrative example is the following. Assume a frequency-flat, time-varying channel. The channel inputs are symbols a_k, $k = 0, \ldots, N_s - 1$. The channel outputs are r_k, $k = 0, \ldots, N_s - 1$, with

$$r_k = \alpha_k a_k + n_k$$

where $n_k \sim \mathcal{CN}\left(0, 2\sigma^2\right)$. The a priori distribution of $\boldsymbol{\alpha} = \left[\alpha_0, \ldots, \alpha_{N_s-1}\right]^T$, $p\left(\boldsymbol{\alpha}\right)$, is known. MAP estimation of $\boldsymbol{\alpha}$ leads to

$$\hat{\boldsymbol{\alpha}}_{MAP} = \arg\max_{\boldsymbol{\alpha}} p\left(\mathbf{r}\left|\boldsymbol{\alpha}\right.\right) p\left(\boldsymbol{\alpha}\right). \tag{13.1}$$

Different ways of solving (13.1) by exploiting code properties can be conceived. First of all, we can simply apply the EM-based techniques operating on the synchronized factor graph of the factorization of $p\left(\mathbf{a}\left|\mathbf{r}, \hat{\boldsymbol{\alpha}}\right.\right)$. Since the parameter to be estimated contains many components, SAGE-like algorithms are required. Alternatively, overall factor graphs can be constructed that incorporate the unknown parameter $\boldsymbol{\alpha}$, as well as its a priori distribution. Finally, more ad-hoc techniques have been developed. These three types of estimators are currently receiving much attention from the technical community.

EM-based techniques

The most obvious way to proceed, would be to apply the SAGE algorithm to estimate the different α_k's. Unfortunately, although such a technique guarantees convergence to a local maximum of the a posteriori distribution $p\left(\boldsymbol{\alpha}\left|\mathbf{r}\right.\right)$, many local maxima are present. This makes straightforward SAGE estimation of time-varying parameters very sensitive to the initial estimate.

More successful attempts are reported in [136, 137]: in [136] the recursive EM algorithm is applied with fixed-interval Kalman smoothing for tracking fast Rayleigh frequency-flat channels. Kalman smoothing is also used in a similar way in [137] for under-sampled frequency selective fading channels.

Overall Factor Graphs

The use of overall factor graphs for estimating time-varying parameters has received some attention in the last few years. In the context of phase estimation, we mention [102–104]: assuming a simple auto-regressive model, the a priori knowledge about the phase was be incorporated into the overall factor graph. As we explained in Chapter 9, the main problem with this approach lies in the computation and representation of messages. This problem is avoided in [30, 107] for tracking frequency-flat channels, modeled as a first-order Markov chain: the overall factor graph is broken into two parts, one corresponding to the detector, operating under the assumption of perfect channel knowledge, and a second part corresponding to the channel, operating under the assumption of perfect knowledge of the transmitted sequence. The part of the factor graph corresponding to the channel then accepts decisions (either hard or soft) from the part corresponding to the decoder. The part corresponding to the decoder, accepts an estimate of the channel coefficients. Within both parts, the standard sum-product algorithm is applied. For the part of the graph corresponding to the channel, this turns out to be equivalent to standard Kalman smoothing. As is shown in [8], the corresponding messages can be represented and computed in a very simple way.

Ad-hoc methods

While the EM-based techniques as well as overall factor graphs have the benefit of providing near-optimal solutions (in some sense), they have a common drawback: they require explicit knowledge of the a priori distribution of the channel parameters. When such knowledge is not available, a mismatch occurs between the channel model and the actual channel, which in turn will degrade the performance [138].

Many other code-aided estimators have been described, which do not require exact knowledge of this a priori distribution, and are therefore inherently sub-optimal. Some of these estimators require significant modifications to the detector (e.g., techniques based on per-survivor processing [139], and those from [140, 141]).

An important class are the augmented estimators. These are conventional estimators (i.e., from before the turbo-era), that have been augmented[1] to accept soft information from a detector/decoder. This idea was applied in [142] for timing recovery and in [143] for phase noise estimation. In both cases, a Phase-Locked Loop (PLL) is used, whereby the conventional hard decisions are replaced with soft decisions.

13.2.3 Discussion

When designing code-aided estimation algorithms, care must be taken to develop algorithms that (i) do not cause too much computational overhead, (ii) do not require severe modifications to the detector. In both respects, embedded EM-based estimation techniques seem the most promising for estimating static parameters.

As far as code-aided estimation of time-varying parameters is concerned, the race is still very much open. Factor graphs are attractive from a theoretical point of view, but may be too complex to implement and require significant

[1]much like the MMSE equalizer we described in Chapter 5.

modifications to existing, off-the-shelf receivers. EM-based techniques seem to offer a good performance-complexity trade-off, but a general framework is still absent. For very specific problems, such as phase noise estimation and timing jitter, more ad-hoc approaches may be the best solution. In any case, it would seem that the most promising approaches to perform code-aided estimation will be a combination of existing techniques (PLLs, Kalman Filters, the EM algorithm etc.), and new insights (such as factor graphs and the sum-product algorithm).

13.3 Link adaptation

Already briefly mentioned in Chapter 6, the problem of link adaptation has been studied extensively in the last few years. See [13, 20, 144] and references therein. The idea behind link adaptation is to define a strategy to dynamically adapt the transmit mode, in order to increase both the spectral efficiency and the data rate. For instance, when information cannot be conveyed reliably over the channel in a given transmit mode (e.g., when the channel is in a deep fade), a different code and/or modulation scheme may be employed.

This approach to link adaptation brings to mind the following problem: for the transmitter to adapt to the channel conditions, it needs to have some kind of knowledge regarding the current channel state. This implies that the receiver must send back channel state information to the transmitter. Hence, there is a strong link between link adaptation and channel estimation.

Information-theoretic approaches to the impact of channel knowledge at the transmitter and the receiver were treated in [145]. With the advent of multi-antenna (MIMO) systems, with their enormous potential for capacity gains, this issue is again receiving increased attention. A recent overview on this topic can be found in [146]. Many aspects need to be considered: what type of channel state information will be sent back to the receiver, long-term or short-term information, how often, with what delay? In theory, when both transmitter and receiver have perfect channel knowledge, the transmitter can often be designed in such a way that the task of the receiver becomes much more simple. For instance, a MIMO system can be converted into a number of parallel, non-interfering systems, making detection a very simple task of low computational complexity [147]. In practice, one often assumes that only the receiver has perfect channel state information. This makes the receiver much more difficult to implement, with a very high computational cost [148].

We conclude that channel estimation and link adaptation are closely related.

13.4 Main Points

The estimation algorithms proposed in Part III of this dissertation can (and have been) extended to a wide variety of estimation and synchronization problems in quasi-static environments. In order to perform code-aided estimation of time-varying parameters, no best solution has yet been found. We have touched upon some important links between the problem of estimating the channel state and link adaptation. A unified theory encompassing code-aided estimation algorithms, channel state feedback, link adaptation and the impact of the complexity of the transmitter and the receiver are important topics for future work, especially for time-varying channels.

Chapter 14

Conclusions

14.1 Introduction

In this dissertation, we have striven to design a receiver, suitable for coded multi-mode transmission in a quasi-static block-fading environment. In multi-mode transmission, the transmitter can select one of several transmit modes, depending on channel conditions, content type etc. The transmit mode is completely defined by a set of parameters. These may include the code, code rate, modulation format, bandwidth, transmission rate, transmit power, etc. Both the transmitter and the receiver are required to adapt to the transmit mode in an efficient way. In general, the receiver has no exact knowledge of the current transmit mode and must perform mode detection. The receiver also faces the challenges present in mono-mode receivers: namely those of channel estimation, delay estimation, synchronization, equalization, and demodulation. On top of all these problems, we have assumed coded transmission, whereby the data-stream is protected by a powerful error-correcting code (such as a turbo- or an LDPC code). Although such codes are very promising in a sense that they achieve performance near the Shannon limit, they operate at extremely low signal-to-noise-ratios (SNR). This makes the tasks of estimation very difficult.

The goal of this thesis was two-fold: on the one hand we wanted to develop a detector that can efficiently *adapt* to varying channel conditions and transmit modes. On the other hand, we have derived a class of code-aided *estimation* algorithms that may be used in low-SNR environments. Of course, the detector and the estimator have to be designed to cooperate in a seamless way.

14.2 Adaptation

Capitalizing on the concepts from Factor Graphs and the Sum-Product Algorithm, we have described several iterative detectors. All these detectors require the conversion of the received continuous-time signal to a suitable discrete-time observation model, from which a factor graph is constructed. To keep the factor graphs simple, the observation model assumes exact knowledge of the channel state, synchronization parameters and transmit mode. Applying the sum-product algorithm on the resulting factor graph results in the computation of the a posteriori probabilities (APPs) of the coded bits, the information bits, the coded symbols, etc. The transmitted information stream can then be recovered by making decisions based on these APPs.

When the channel conditions change, part of this factor graph may have to be altered to accommodate these changes. The same is true for mode adaptation: when a new transmit mode is selected, the corresponding part in the factor graph is altered. As a factor graph is an abstract notion that would be implemented in software in the receiver, adaptation to channel state and transmit mode are achieved by simply loading the appropriate software. One important exception is the symbol rate: a change in the symbol rate translates into a change of the conversion from the continuous-time signal to the discrete-time observation model, and cannot easily be captured in the factor graph framework. This conversion should be implemented as efficiently as possible. Since multi-rate transmission is an important feature in current- and next-generation communication standards, we have examined in great detail several low-complexity multi-rate receivers. Our main conclusion was that such a receiver could be constructed so that

- the IF signal can be sampled directly, removing the need for identical analog in-phase an quadrature components;
- most digital filters operate at a rate that is *independent* of the symbol rate;
- the corresponding BER degradation is negligible (less than 0.1 dB).

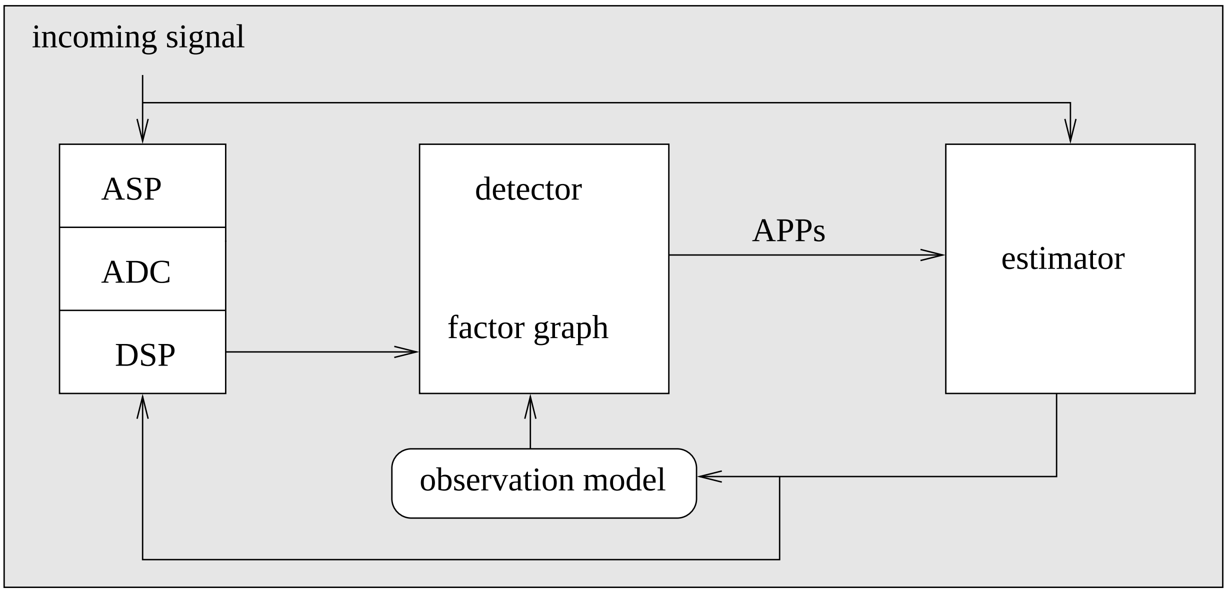


Figure 14.1: *Detector and Estimator: through analog signal processing (ASP) and digital signal processing (DSP) the received signal is converted to a suitable observation. A model of this observation is cast into a factor graph. The estimator updates parameters of this observation model, based on a posteriori probabilities (APPs) computed on the factor graph.*

Such a receiver could be applied to Variable Chip Rate (VCR) DS/SS systems, and as such, removes one of the important drawbacks of VCR-transmission, namely the complexity of the receiver.

14.3 Estimation

The abovementioned detectors all rely on the knowledge of the channel state, synchronization parameters and transmit mode. In practice, some of these parameters will have to be estimated by the receiver. Developing estimation algorithms was the second main topic of this thesis. These estimation algorithms should be of relatively low complexity, should not require significant modifications to the existing detector, have short acquisition times, and be very reliable, even at low SNR. Rather than developing ad-hoc algorithms, we have taken a more systematic approach and started from the Maximum Likelihood (ML) principle. Since ML estimation is not tractable in practice, we have considered estimation on factor graphs, the Expectation-Maximization (EM) and the Space-Alternating Expectation Maximization (SAGE) algorithms as a means to perform ML estimation in an iterative and computationally attractive way. From these very general techniques, we have derived estimation algorithms for our coded multi-mode receiver that iterate between data detection on factor graphs and parameter estimation. A diagram depicting this interaction is shown in Fig. 14.1.

In its pure form, these estimation algorithms are outrageously complex. To alleviate this, we introduced several ways to reduce the complexity without significantly impacting the performance. Based on a study of the convergence properties of the EM-based estimation algorithms, we have defined two variations: the parallel EM algorithm and the discrete EM algorithm. The former can be applied when no reliable initial estimate is available, while the latter has applications in problems dealing with the determination of discrete parameters (e.g., frame synchronization, mode detection).

These code-aided estimation algorithms have been applied successfully to the estimation of both frequency-selective and frequency-flat channels, to the estimation of both continuous and discrete parameters, and to the estimation of transmit modes. Furthermore, the algorithms can easily be applied to different estimation problems in quasi-static environments.

14.4 Publications

The research described in this dissertation has led to the following publications in refereed international conferences and journals.

Factor graphs and the sum-product algorithm [149–153]

Low-complexity receivers for Software Radio [73,75,77,154–156]

Code-aided estimation

- Frequency-flat channels: [110,127,157–163]
- Frequency-selective channels: [125,131–133,164–168]
- Discrete Parameters: [169–173]
- Mode detection: [124,174]

Bibliography

[1] Politis, C.; Oda, T.; Dixit, S.; Schieder, A.; Lach, H.-Y.; Smirnov, M.I.; Uskela, S.; Tafazolli, R. "Cooperative networks for the future wireless world". *IEEE Communications Magazine*, 42(9):70–79, September 2004.

[2] Asatani, K.; Bigi, F.; Probst, P.-A. "Telecommunications standardization for the new millennium: ITU-T\'s strategies". *IEEE Communications Magazine*, 37(4):124–130, April 2001.

[3] J.M. Wozencraft and I.M. Jacobs. *Principles of Communications Engineering*. John Wiley & Sons, 1967.

[4] C.E. Shannon. "A mathematical theory of communication". *Bell System Technical Journal*, 27:379–423, July 1948.

[5] S.-Y. Chung, G.D. Forney, T.J. Richardson and R. Urbanke. "On the desing of low-density parity check codes within 0.0045 dB of the Shannon limit". *IEEE Communications Letters*, 5(2):58–60, February 2001.

[6] R.G. Gallager. "Low density parity-check codes". *IRE Trans. Inform. Theory*, IT-8:pp. 21–29, Jan. 1962.

[7] C. Berrou, A. Glavieux and P. Thitimajsima. "Near Shannon limit error-correcting coding and decoding: Turbo Codes". In *Proc. IEEE International Conference on Communications (ICC)*, pages 1064–1070, Geneva, Switzerland, May 1993.

[8] F. Kschischang, B. Frey and H.-A. Loeliger. "Factor graphs and the sum-product algorithm". *IEEE Trans. Inform. Theory*, 47(2):pp.498–519, February 2001.

[9] S. Aji and R. McEliece. "The generalized distributive law". *IEEE Transactions on Information Theory*, 46:325–353, March 2000.

[10] Seong Taek Chung and A.J. Goldsmith. "Degrees of freedom in adaptive modulation: a unified view". *IEEE Trans.. on Communications*, 49(9):1561–1571, September 2001.

[11] J.F. Hayes. "Adaptive feedback communications". *IEEE Trans. on Communications*, 16(1):71–81, February 1968.

[12] J. Hancock and W. Lindsey. "Optimum performance of self-adaptive systems operating through a Rayleigh-fading medium". *IEEE Trans. on Communications*, 11(4):443–453, December 1963.

[13] Mallik, R.K.; Win, M.Z.; Shao, J.W.; Alouini, M.-S.; Goldsmith, A.J. "Channel capacity of adaptive transmission with maximal ratio combining in correlated Rayleigh fading". *IEEE Trans. on Wireless Communications*, 3(4):1124–1133, July 2004.

[14] J. Mitola. "The Software Radio Architecture". *IEEE Comm. Mag.*, 30(5):26–38, May 1995.

[15] T. Hentschel. *Sample rate conversion in software configurable radios*. Artech House, Boston, London, 2002.

[16] H. Meyr, M. Moeneclaey, S.A. Fechtel. *Synchronization, Channel Estimation, and Signal Processing*, volume 2 of *Digital Communication Receivers*. John Wiley & Sons, 1997.

[17] U. Mengali and A.N. D'Andrea. *Synchronization Techniques for Digital Receivers*. Plenum Press, 1997.

[18] A. Glavieux, C. Laot and J. Labat. "Turbo equalization over a frequency selective channel". In *Proceedings of the International Symposium on Turbo Codes*, pages 96–102, Brest, France, September 1997.

[19] N. Noels, C. Herzet, A. Dejonghe, V. Lottici, H. Steendam, M. Moeneclaey, M. Luise and L. Vandendorpe. "Turbo-synchronization: an EM algorithm interpretation". In *Proc. IEEE International Conference on Communications (ICC)*, Anchorage, May 2003.

[20] S. Vishwanath and A. J. Goldsmith. "Adaptive Turbo Coded Modulation for Flat Fading Channels". *IEEE Trans. on Communications*, 51(6):964–972, June 2003.

[21] J.G. Proakis. *Digital Communications*. McGraw-Hill, 4th edition, 2001.

[22] J.E. Brittain. "The legacy of Edwin Howard Armstrong". *Proceedings of the IEEE*, 79(2), 1991.

[23] G.D. Forney. "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference". *IEEE Trans. on Information Theory*, 18(3):363–378, May 1972.

[24] R. Koetter, A.C. Singer and M. Tüchler. "Turbo equalization". *Signal Processing Magazine*, 21(1):67–80, January 2004.

[25] D. Raphaeli and Y. Zarai. "Combined turbo equalization and turbo decoding". In *Proceedings of the International Symposium on Turbo Codes*, pages 180–183, Brest, France, Sept. 1997.

[26] A. M. Guidi and L. P. Sabel. "Digital demodulator architectures for bandpass sampling receivers". *Proceedings of the 7th International Tyrrhenian Workshop on Digital Communications*, pages 183–194, 1995.

[27] S. Buzzi and A. De Maio. "Code-aided blind rate detection for multirate DS/CDMA". In *Proc. IEEE 7th International Symposium on Spread Spectrum Techniques and Applications*, pages 19–23, vol. 1, 2002.

[28] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.

[29] G.D. Forney. "Codes on Graphs: Normal Realizations". *IEEE Trans. on Information Theory*, 47(2):520–545, February 2001.

[30] Manyuan Shen, Huaning Niu and Hui Liu. "Iterative receiver design in Rayleigh fading using factor graph". In *Proc. IEEE Vehicular Technology Conference (VTC Spring)*, Hong-Kong, April 2003.

[31] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". *IEEE Trans. on Information Theory*, 13(2):260–269, April 1967.

[32] X. Li, A. Chindapol and J.A. Ritcey. "Bit-interleaved coded modulation with iterative decoding and 8PSK signaling". *IEEE Trans. on Comm.*, 50(8):1250–1257, August 2002.

[33] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv. "Optimal decoding of linear codes for minimising symbol error rate". *IEEE Trans. Inform. Theory*, 20:pp. 284–287, March 1974.

[34] D.J.C. MacKay. "Good error-correcting codes based on very sparse matrices". *IEEE Trans. Inform. Theory*, 45(2):pp. 399–431, March 1999.

[35] X. Li and J.A. Ritcey. "Trellis-coded modulation with bit interleaving and iterative decoding". *IEEE Journal on Selected Areas in Comm.*, 17(4), April 1999.

[36] J.J. Boutros, F. Boixadera and C. Lamy. "Bit-interleaved coded modulations for multiple-input multiple-output channels". In *Proceedings of the IEEE 6th Int. Symp. on Spread-Spectrum Tech. & Appli.*, pages 123–126, New Jersey, USA, September 2000.

[37] H.-A. Loeliger. "An introduction to factor graphs". *IEEE Signal Processing Magazine*, 21(1):28–41, January 2004.

[38] J. Hagenauer and P. Hoeher. "A Viterbi algorithm with soft-decision outputs and its applications". In *Proc. IEEE Globecom*, pages 1680–1686, vol. 3, November 1989.

[39] F. Gardner. "Interpolation in digital modems - Part I: Fundamentals". *IEEE Trans. Comm.*, 41(3):501–507, 1993.

[40] F. Gardner, L. Erup and R.A. Harris. "Interpolation in Digital Modems, Part II: implementation and performance". *IEEE Trans. Comm.*, 41(6):pp. 998–1008, 1993.

[41] C.W. Farrow. "A Continuously Variable Digital Delay Element". In *Proc. IEEE International Symposium on Circuits and Systems*, pages 2641–2645, Espoo, Finland, June 1988.

[42] E.B. Hogenauer. "An economical class of digital filters for decimation and interpolation". *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-29(2):155–162, April 1981.

[43] W. A. Abu-Al-Saud and G.L. Stüber. "Modified CIC filter for sample rate conversion in software radio systems". *Proc. IEEE Signal Processing Letters*, 10(5):152–154, May 2003.

[44] K. Bucket and M. Moeneclaey. "BER degradation caused by non-ideal interpolation of bandlimited DS/SS signals". *AEÜ International Journal of Electronics and Communications*, 48(5):231–236, 1994.

[45] K. Bucket and M. Moeneclaey. "The effect of interpolation on the BER performance of narrowband BPSK and (O)QPSK on rician fading channels". *IEEE Trans. Comm.*, 42(11):2929–2933, 1994.

[46] H.L. Van Trees. *Detection, Estimation,and Modulation Theory, Part I.* Wiley and Sons, October 2001.

[47] R.W. Lucky. "Automatic equalization for digital communications". *Bell Syst. Tech. Journal*, 44:547–588, April 1965.

[48] M.E. Austin. "Decision-feedback equalization for digital communication over dispersive channels". Technical Report 437, MIT Lincoln Library, Lexington, Mass., August 1967.

[49] X. Wang and H. V. Poor. "Iterative (turbo) soft interference cancellation and decoding for coded CDMA". *IEEE Trans. Comm.*, 47(7):pp.1046–1061, July 1999.

[50] J. Hagenauer, E. Offer, C. Méasson and M. Mörz. "Decoding and equalization with analog non-linear networks". *European Trans. Comm.*, 10:pp.659–680, November 1999.

[51] M. Tüchler, R. Koetter and A.C. Singer. "Turbo-equalization: principles and new results". *IEEE Trans. on Communications*, 50(5):754–767, May 2002.

[52] Zi-Ning Wu and J.M. Cioffi. "Low-complexity iterative decoding with decision-aided equalization for magnetic recoding channels". *IEEE Journal on Selected Areas in Communications*, 19(4):699–708, 2001.

[53] J. Hagenauer. "Rate-compatible puncture convolutional codes (RCPC codes) and their applications". *IEEE Trans. on Communications*, 36(4):389–400, April 1988.

[54] A.S. Barbulescu and S.S. Pietrobon. "Rate-compatible turbo codes". *Electronics Letters*, 31(7):535–536, March 1995.

[55] A.J. Goldsmith and S. Chua. "Adaptive coded modulation for fading channels". *IEEE Trans. on Comm.*, 46(5):595–602, May 1998.

[56] M.-S. Alouinim X. Tang and A.J. Goldsmith. "Adaptive modulation scheme for simultaneous voice and data transmission over fading channels". *IEEE Journal on Selected Areas in Communications*, 75(5):837–850, 1999.

[57] T. Keller and L. Hanzo. "Adaptive modulation techniques for duplex OFDM transmission". *IEEE Transactions on Vehicular Technology*, 49(5):1893–1906, September 2000.

[58] Ue, T.; Sampei, S.; Morinaga, N.; Hamaguchi, K. "Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission". *IEEE Trans. on Vehicular Technology*, 47(4):1134–1147, November 1998.

[59] Shengli Zhou and G.B. Giannakis. "Adaptive Modulation for multiantenna transmissions with channel mean feedback". *IEEE Trans. on Wireless Communications*, 3(5):1626–1636, September 2004.

[60] S. Roy and H. Yan. "Blind channel estimation in multi-rate CDMA systems". *IEEE Trans. on Communications*, 50(6):995–1004, June 2002.

[61] T. Ottosson and A. Svensson. "Multi-rate schemes in DS/CDMA". In *Proc. IEEE Vehicular Technology Conference*, pages 1006–1010, Chicago, July 1995.

[62] Srinivasan, R.; Mitra, U.; Moses, R.L. "Design and analysis of receiver filters for multiple chip-rate DS-CDMA systems". *IEEE Journal on Selected Areas in Communications*, 17(12):2096–2109, December 1999.

[63] Buzzi, S.; Lops, M.; Tulino, A.M. "Blind adaptive multiuser detection for asynchronous dual-rate DS/CDMA system". *IEEE Journal on Selected Areas in Communications*, 19(2):233–244, February 2001.

[64] E. Buracchini. "The Software Radio Concept". *IEEE Comm. Mag.*, 38(9):138–143, September 2000.

[65] T. Hentschel and G. Fettweis. "Sample Rate Conversion for Software Radio". *IEEE Comm. Mag.*, 38(8):142–150, August 2000.

[66] C.L. Liu. "Impacts of I/Q imbalance on QPSK-OFDM-QAM detection". *IEEE Trans. on Consumer Electronics*, 44(3):984–989, August 1998.

[67] K. Bucket and M. Moeneclaey. "Digital modems using non-synchronized sampling: matched filter + interpolator versus interpolator + matched filter". In *Proc. IEEE Signal Processing Symposium, SPS98*, pages 31–34, Leuven,Belgium, March 1998.

[68] H.G. Göckler, G. Evangelista and A. Groth. "Minimal block processing approach to fractional sample rate conversion". *Signal Processing*, 81(4):673–691, April 2001.

[69] M. Henker and G. Fetweiss. "Combined Filter for Sample Rate Conversion, Matched Filtering and Symbol Synchronization in Software Radio Terminals". In *Proceedings of the European Wireless*, pages 61–66. VDE Verlag Berlin Offenbach, September 2000.

[70] T. Hentschel and G.P. Fetweiss. "Continuous-Time Digital Filters for Sample-Rate Conversion in Reconfigurable Radio Terminals". In *Proceedings of the European Wireless*, pages 55–59. VDE Verlag Berlin Offenbach, September 2000.

[71] L. Lundheim and T.A. Ramstad. "An efficient and flexible structure for decimation and sample rate adaptation in Software Radio receivers". In *Proc. ACTS Mobile Comm. Summit*, pages 663–668, June 1999.

[72] D. Babic, J. Vesma and M. Renfors. "Decimation by irrational factor using CIC filters and linear interpolation". In *Proc. ICASSP'01*, Utah, May 2001.

[73] H. Wymeersch and M. Moeneclaey. "Multi-rate receiver design with IF sampling and digital timing correction". In *Proc. VTC fall*, Orlando, USA, October 2003.

[74] T. Hentschel, M. Henker and G.P. Fetweiss. "The digital front-end of software radio terminals". *IEEE Personal Communications*, 6(4):40–46, August 1999.

[75] H. Wymeersch and M. Moeneclaey. "Low complexity multi-rate IF sampling receivers using CIC filters and polynomial interpolation". In *Proc. Sixth Baiona Workshop on Signal Processing in Communications*, Baiona, Spain, September 2003.

[76] M.P. Donadio. "CIC filter introduction". 2000.
http://www.dspguru.com/info/tutor/cic.htm.

[77] H. Wymeersch and M. Moeneclaey. "BER performance of software radio multirate receivers with nonsynchronized IF sampling and digital timing correction". In *Proc. ICASSP'03*, Hong Kong, April 2003.

[78] C. Mosquera and M. Cacheda. "Feedback timing synchronization for IF-sampled systems". In *Proc. 8th International Workshop on Signal Processing for Space Communications, SPSC 2003*, 2003.

[79] S. Mollenkopf. "Aperture jitter in IF sampling CDMA receivers". In *Proc. IEEE MTT-S Symposium on Technologies for Wireless Applications*, pages 69–72, February 1997.

[80] Hae-Moon Seo; Chang-Gene Woo; Pyung Choi. "Relationship between ADC performance and requirements of digital-IF receiver for WCDMA base-station". *IEEE Trans. on Vehicular Technology*, 52(5):1398–1408, September 2003.

[81] 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*, 39(1):pp. 1–38, 1977. Series B.

[82] C. F. J. Wu. "On the convergence properties of the EM algorithm". *The Annals of Statistics*, 11(1):95–109, 1983.

[83] Gallo, A.S.; Vitetta, G.M.; Chiavaccini, E. "A BEM-based algorithm for soft-in soft-output detection of co-channel signals". *IEEE Transactions on Wireless Communications*, 3(5):1533–1542, September 2004.

[84] J.A. Fessler and A.O. Hero. "Space-alternating generalized expectation-maximization algorithm". *IEEE Trans. Signal Processing*, 42(10):2664–2677, Oct. 1994.

[85] A.W. Eckford. "Channel estimation in block fading channels using the factor graph EM algorithm". In *Proc. 22nd Biennial Symposium on Communications*, Kingston, Canada, 2004.

[86] M. Feder and E. Weinstein. "Parameter estimation of superimposed signals using the EM algorithm". *IEEE Trans. on Acoustics, Speech and Signal Processing*, 36:477–489, April 1988.

[87] B.H. Fleury, M. Tschudin, R. Heddergott, D. Dahlhaus and K.I. Pedersen. "Channel parameter estimation in mobile radio environments using the SAGE algorithm". *IEEE Journal on Selected Areas in Communications*, 17(3):pp.434–450, March 1999.

[88] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley and Sons, Inc, New York, 1991.

[89] M. Morelli, A. D'Andrea and U. Mengali. "Feedforward ML-based timing estimation with PSK signals". *IEEE Trans. Comm.*, 1(3):80–82, May 1997.

[90] M. Oerder and H. Meyr. "Digital filter and square timing recovery". *IEEE Trans. on Comm.*, 36:605–612, May 1988.

[91] J.L. Massey. "Optimum frame synchronization". *IEEE Trans. Comm.*, com-20(2):pp. 115–119, April 1972.

[92] A.J. Viterbi and A.M. Viterbi. "Nonlinear estimation of PSK-modulated carrier phase with application to burst digital transmission". *IEEE Trans. Inform. Theory*, IT-29:pp. 543–551, July 1983.

[93] Y. Li, C.N. Georghiades and G. Huang. "Sequence estimation for space-time coded systems". *IEEE Trans. Comm.*, 49(6):948–951, June 2001.

[94] E. Panayirci, U. Aygoly and A.E. Pusane. "Sequence estimation with transmit diversity for wireless communications". *International Journal of Electronics and Communications*, 57(5):309–316, September 2003.

[95] C. N. Georghiades and J. C. Han. "Sequence estimation in the presence of random parameters via the EM algorithm". *IEEE Trans. Comm*, 45(3):pp.300–308, March 1997.

[96] N. Noels, V. Lottici, A. Dejonghe, H. Steendam, M. Moeneclaey, M. Luise , L. Vandendorpe. "A Theoretical Framework for Soft Information Based Synchronization in Iterative (Turbo) Receivers". *EURASIP Journal on Wireless Communications and Networking JWCN, Special issue on Advanced Signal Processing Algorithms for Wireless Communications*, 2005. Accepted for publication.

[97] V. Ramon, C. Herzet, L. Vandendorpe and M. Moeneclaey. "EM algorithm-based multiuser synchronization in turbo receivers". In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, Canada, May 2004.

[98] C. Herzet, V. Ramon, L. Vandendorpe and M. Moeneclaey. "EM algorithm-based timing synchronization in turbo receivers". In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Hong Kong, April 2003.

[99] M.A. Khalighi and J.J. Boutros. "Modified unbiased EM-based channel estimation for MIMO turbo receivers". In *Proc. IEEE Symposium on Signal Processing and Information Technology*, Rome, Italy, December 2004.

[100] A.A. D'Amico, U. Mengali and M. Morelli. "Channel estimation for the Uplink of a DS-CDMA system". *IEEE Trans. on Wireless Comm.*, 2(6):1132–1137, November 2003.

[101] A. Kocian and B.H. Fleury. "EM-based joint data detection and channel estimation of DS-CDMA signals". *IEEE Trans. on Comm.*, 51(10):1709–1720, October 2003.

[102] J. Dauwels and H.-A. Loeliger. "Joint decoding and phase estimation: an exercise in factor graphs". In *Proc. Int. Symposium in Information Theory*, Yokohama, Japan, July 2003.

[103] I. Sutskover, S. Shamai and J. Ziv. "A novel approach to iterative joint decoding and phase estimation". In *Proc. 3rd Int. Symp. on Turbo Codes and Related Topics*, pages 87–90, Brest, France, September 2003.

[104] G. Colavolpe, A. Barbieri, G. Caire and N. Bonneau. "Bayesian and non-Bayesiab methods for joint decoding and detection in the presense of phase noise". In *Proc. IEEE Int. Symposium on Information Theory (ISIT)*, Chicago, USA, July 2004.

[105] K. Sasha, H.-A. Loeliger, A.G. Lindgren. "AR model parameter estimation: from factor graphs to algorithms". In *Proc. IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, Canada, May 2004.

[106] A.P. Worthen and W.E. Stark. "Unified design of iterative receivers using factor graphs". *IEEE Transactions on Information Theory*, 47(2):843–849, February 2001.

[107] T. Wadayama. "An iterative decoding algorithm for channels with additive linear dynamical noise". *IEICE Transactions on Fundamentals*, E86-A(10):2452–2460, October 2003.

[108] David J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[109] V. Lottici and M. Luise. "Embedding carrier phase recovery into iterative decoding of turbo-coded linear modulations". *IEEE Transactions on Communications*, 52(4):661–669, April 2004.

[110] H. Wymeersch and M. Moeneclaey. *Code-aided ML joint delay estimation and frame synchronization*, volume 27 of *Signal Processing for Telecommunications and Multimedia, Multimedia Systems and Applications*, chapter 8, pages 97–110. Springer-Verlag, 2005.

[111] P. Spasojevic and C.N. Georghiades. "On the (non) convergence of the EM algorithm for discrete parameter estimation". In *Proc. Allerton Conference*, Monticello, Illinois, Oct. 2000.

[112] M.K. Howlader and B.D. Woerner. "Decoder-assisted frame synchronization for packet transmission". *IEEE journal on selected areas in comm.*, 19(12):pp. 2331–2345, Dec. 2001.

[113] E. Cacciamani and C. Wolejsza. "Phase-ambiguity resolution in a four-phase PSK communications system". *IEEE Trans. on Comm*, COM-19(6):1200–1210, Dec. 1971.

[114] J. Sodha. "Turbo code frame synchronization". *Signal Processing Journal, Elsevier*, 82:pp. 803–809, 2002.

[115] P. Robertson. "Improving frame synchronization when using convolutional codes". In *Proc. IEEE GLOBECOM*, pages 1606–1611, Houston, 1993.

[116] B. Mielczarek and A. Svensson. "Timing error recovery in turbo coded systems on AWGN channels". *IEEE Trans. on Comm.*, 50(10):pp. 1584–1592, Oct. 2002.

[117] T.M. Cassaro and C.N. Georghiades. "Frame synchronization for coded systems over AWGN channel". *IEEE Trans. on Comm.*, 52(3):484–489, March 2004.

[118] U. Mengali, A. Sandri and A. Spalvieri. "Phase ambiguity resolution in trellis-coded modulations". *IEEE Trans. Comm*, 38(12):2087–2088, Dec. 1990.

[119] A. Sharma and U. Mitra. "Blind rate detection for multirate UMTS DS-CDMA signals". In *Proc. IEEE International Conference on Communications (ICC'01)*, pages 2504–2509, vol. 8, Helsinki, Finland, June 2001.

[120] A. Gutierrez, Y.K. Lee and G. Mandyam. "A ML rate detection algorithm for IS-95 CDMA". In *IEEE Vehicular Technology Conference (VTC Spring'99)*, pages 417–421, vol. 1, May 1999.

[121] P. Ciblat, P. Loubaton, E. Serpedin and G.B. Giannakis. "Asymptotic Analysis of blind cyclic correlation-based symbol-rate estimators". *IEEE Trans. on Information Theory*, 48(7):1922–1934, July 2002.

[122] A.V. Dandawaté and G.B. Giannakis. "Statistical tests for presence of cyclostationarity". *IEEE Trans. on Signal Processing*, 42:2355–2369, September 1994.

[123] Jong Youl Lee, Young Mo Chung, and Sang Uk Lee. "On a timing recovery technique for a variable symbol rate signal". In *Proc. IEEE Vehicular Technology Conference*, pages 1724–1728, Phoenix, AZ, USA, May 1997.

[124] H. Wymeersch and M. Moeneclaey. "ML-based blind symbol rate detection for multi-rate receivers". In *Proc. IEEE International Conference on Communications (ICC05)*, Seoul, Korea, May 2005.

[125] M. Guenach, H. Wymeersch and M. Moeneclaey. "Joint estimation of path delay and complex gain for coded systems using the EM algorithm". In *Proc. International Zürich Seminar (IZS'04)*, Zürich, Switserland, February 2004.

[126] J. Campello and D.S. Modha. "Extended bit-filling and LDPC code design". In *Proc. Globecom*, San Antonio, USA, November 2001.

[127] J. Dauwels, H. Wymeersch, H.-A. Loeliger and M. Moeneclaey. "Phase Estimation and Phase Ambiguity Resolution by Message Passing". *Lecture Notes in Computer Science, Springer-Verlag*, 3124:150–155, 2004.

[128] M. Moeneclaey. "On the true and the modified Cramer-Rao bounds for the estimation of a scalar parameter in the presence of nuisance parameters". *IEEE Trans. Comm.*, 46(11):1536–1544, November 1998.

[129] G. Caire, G. Taricco and E. Biglieri. "Bit-interleaved coded modulation". *IEEE Trans. on Information Theory*, 44:927–946, May 1998.

[130] E. Zehavi. "8-PSK trellis codes for Rayleigh fading channels". *IEEE Trans. on Comm.*, 41:873–883, May 1992.

[131] M. Guenach, H. Wymeersch and M. Moeneclaey. "Iterative joint timing and carrier phase estimation using the SAGE algorithm for a coded DS-CDMA system". In *Proc. International Symposium on Control, Communications and Signal Processing (ISCCSP)* , Hammamet, Tunisia, March 2004.

[132] M. Guenach, H. Wymeersch and M. Moeneclaey. "On DS-CDMA uplink channel parameter estimation for bit-interleaved-coded modulation". In *Proc. 2nd International Workshop on Signal Processing for Wireless Communications (SPWC'04)*, London, UK, June 2004.

[133] M. Guenach, H. Wymeersch and M. Moeneclaey. "On soft multiuser channel estimation of DS-CDMA uplink using different mapping strategies". In *Proc. IEEE International Conference on Communications (ICC)*, Seoul, Korea, May 2005.

[134] M. Guenach, F. Simoens and M. Moeneclaey. "Parameter estimation in a space-time bit-interleaved coded modulation scheme for DS-CDMA". In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, USA, March 2005.

[135] Z. Baranski, A.M. Haimovich and J. Garcia-Frias. "EM-based iterative receiver for space-time coded modulation with noise variance estimation". In *Global Telecommunications Conference*, pages 355–359, November 2002.

[136] M. Yan, B.D. Rao. "Soft decision-directed MAP estimate of fast Rayleigh flat fading channels". *IEEE Transactions on Comm.*, 51(12):1965–1969, December 2003.

[137] M. Nissilä and S. Pasupathy. "Adaptive Bayesian and EM-based detectors for frequency-selective fading channels". *IEEE Transactions on Comm.*, 51(8):1325–1336, August 2003.

[138] C. Komninakis, C. Fragouli, A.H. Sayed and R.D. Wesel. "Multi-input multi-output fading channel tracking and equalization using kalman estimation". *IEEE Trans. on Signal Processing*, 50(5), May 2002.

[139] R. Raheli, A. Polydoros, C.-K. Tzou. "Per-survivor processing: a general approach to MLSE in uncertain environments". *IEEE Trans. on Communications*, 43(2):354–364, February 1995.

[140] A. Anastasopoulos and K.M. Chugg. "Adaptive iterative detection for phase tracking turbo-coded systems". *IEEE Transactions on Communications*, 49(12):2135–2144, December 2001.

[141] B. Mielczarek and A. Svensson. "Phase offset estimation using enhanced turbo decoders". In *Proc. IEEE International Conference on Communications (ICC)*, 2002.

[142] J.R. Barry, A. Kavcic, S.W. McLaughlin, A. Nayak and Wei Zeng. "Iterative timing recovery". *IEEE Signal Processing Magazine*, 21(1):89–102, January 2004.

[143] N. Noels, H. Steendam and M. Moeneclaey. "A maximum-likelihood based feedback carrier synchronizer for turbo-coded systems". In *Proc. IEEE Vehicular Technology Conference (VTC Spring)*, Stockholm, Sweden, May 2005.

[144] S. Catreux, V. Erceg, D. Gesbert, and R. W. Heath, Jr. "Adaptive Modulation and MIMO Coding for Broadband Wireless Data Networks". *IEEE Communications Magazine*, pages 108–115, June 2002.

[145] E. Biglieri, J. Proakis and S. Shamai. "Fading channels: information-theoretic and communications aspects". *IEEE Trans. on Information Theory*, 44(6):2619–2692, October 1998.

[146] A. Goldsmith, S.A. Jafar, N. Jindal and S. Vishwanath. "Capacity limits of MIMO channels". *IEEE Journal on selected areas in Communications*, 21(5):684–702, June 2003.

[147] D. Gesbert, M. Shafi, D. Shiu, P.J. Smith and A. Naguib. "From theory to practice: an overview of MIMO space-time coded wireless systems". *IEEE journal on Selected Areas in Comm.*, 21(3):281–302, April 2003.

[148] H. Viklalo, B. Hassibi and T. Kailath. "Iterative decoding for MIMO channels via modified sphere decoding". *IEEE Trans. on Wireless Communications*, 3(6), November 2004.

[149] H. Wymeersch, H. Steendam and M. Moeneclaey. "Interleaved coded modulation for non-binary codes: a factor graph approach". In *Proc. IEEE Global Telecommunications Conference (Globecom'04)*, Dallas, USA, November 2004.

[150] H. Wymeersch, H. Steendam and M. Moeneclaey. "Computational complexity and quantization effects of decoding algorithms of LDPC codes over $GF(q)$". In *Proc. ICASSP*, Montreal, Canada, May 2004.

[151] H. Wymeersch, H. Steendam and M. Moeneclaey. "Log-domain decoding of LDPC codes over $GF(q)$". In *Proc. IEEE International Conference on Communications (ICC)*, June 2004.

[152] F. Simoens, H. Wymeersch and M. Moeneclaey. "Spatial mapping for MIMO systems". In *Proc. Information Theory Workshop (ITW04)*, San Antonio, USA, October 2004.

[153] F. Simoens, H. Wymeersch, H. Bruneel and M. Moeneclaey. "Multi-dimensional mapping for bit-interleaved coded modulation with BPSK/QPSK signaling". *IEEE Communications Letters*, 9(5):453–455, May 2005.

[154] H. Wymeersch and M. Moeneclaey. "The impact of the intermediate frequency selection on the performance of an all-digital bandpass receiver with interpolator". In *Proc. IEEE Benelux Symposium on Communications and Vehicular Technology*, Louvain-La-Neuve, Belgium, October 2002.

[155] H. Wymeersch and M. Moeneclaey. "Multi-rate receivers with IF sampling and digital timing correction". *Elsevier Signal Processing*, 81(11):2067–2079, November 2004.

[156] H. Wymeersch and M. Moeneclaey. "Blind symbol rate detection for low-complexity multi-rate receivers". In *Proc. IEEE Vehicular Technology Conference (VTC Spring)*, Stockholm, Sweden, June 2005.

[157] N. Noels, H. Wymeersch, H. Steendam and M. Moeneclaey. "True Cramer-Rao bound for timing recovery from a bandlimited linearly modulated waveform with unknown carrier phase and frequency". *IEEE Transactions on Communications*, 52(3):473–483, March 2004.

[158] F. Simoens, H. Wymeersch, H. Steendam and M. Moeneclaey. *Synchronization for MIMO systems*, volume Smart Antennas - State-of-the-art, chapter 6. 2005.

[159] F. Simoens, H. Wymeersch, N. Noels, H. Steendam and M. Moeneclaey. "Turbo channel estimation for bit-interleaved coded modulation". In *XI National Symposium of Radio Science (invited paper - NEWCOM session)*, Poznan, Poland, April 2005.

[160] C. Herzet, H. Wymeersch, M. Moeneclaey and L. Vandendorpe. "on maximum-likelihood timing synchronization". *IEEE Trans. on Comm.*, 2005. Accepted.

[161] H. Wymeersch, N. Noels, H. Steendam and M. Moeneclaey. "Synchronization at low SNR: performance bounds and algorithms".
Invited presentation at the Communication Theory Workshop, Capri, Italy, May 2004. Available at http://telin.ugent.be/∼hwymeers/publications.html.

[162] H. Wymeersch and M. Moeneclaey. "Iterative code-aided ML phase estimation and phase ambiguity resolution". *EURASIP Journal on Applied Signal Processing, Special Issue on Turbo Processing*, 2005(6), May 2004.

[163] H. Wymeersch, F. Simoens and M. Moeneclaey. "Code-aided joint channel estimation and frame synchronization for MIMO systems". In *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Lisbon, Portugal, July 2004.

[164] M. Guenach, F. Simoens, H. Wymeersch and M. Moeneclaey. "Frequency offset and channel parameters estimation for a multi-user DS-CDMA system using the code-aware SAGE algorithm". In *Proc. first annual IEEE Benelux/DSP Valley Signal Processing Symposium*, Antwerp, Belgium, April 2005.

[165] M. Guenach, H. Wymeersch and M. Moeneclaey. "On channel parameter estimation in a space-time bit-interleaved coded modulation systems for multi-path DS-CDMA uplink with receive diversity". *IEEE Trans. on Vehicular Technology*, 2005. accepted, available at http://telin.ugent.be/∼hwymeers/publications.html.

[166] M. Guenach, H. Wymeersch, H. Steendam and M. Moeneclaey. "Code-aided ML joint frame synchronization and channel estimation for downlink MC-CDMA". *IEEE Journal on Selected Areas in Communications*, 2005. Submitted.

[167] M. Guenach, F. Simoens, H. Wymeersch and M. Moeneclaey. "Code-aided joint channel and frequency estimation for a ST-BICM multi-user DS-CDMA system". In *Proc. Eusipco*, Antalya, Turkey, September 2005.

[168] H. Wymeersch M. Guenach, F. Simoens and M. Moeneclaey. "Code-aided Joint Channel and Frequency Offset Estimation for DS-CDMA". *IEEE Journal on Selected Areas in Comm.*, 2005. Accepted for publication in the special issue of Next Generation CDMA Technologies.

[169] H. Wymeersch and M. Moeneclaey. "Code-aided frame synchronizers for AWGN channels". In *Proc. International Symposium on Turbo Codes & related topics*, Brest, France, September 2003.

[170] H. Wymeersch, H. Steendam and M. Moeneclaey. "Analysis of an ML data-aided phase ambiguity resolution algorithm for M-PSK". In *Proc. IEEE symposium on Comm. and Vehicular Technology*, Ghent, Belgium, November 2004.

[171] H. Wymeersch, H. Steendam, H. Bruneel and M. Moeneclaey. "Code-aided frame synchronization and phase ambiguity resolution". *IEEE Transactions on Signal Processing*, 2005. Accepted for publication.

[172] H. Wymeersch and M. Moeneclaey. "ML frame synchronization for turbo and LDPC codes". In *Proc. 7th Int. Symp. on DSP and Comm. Systems*, Coolangatta, Australia, December 2003.

[173] H. Wymeersch and M. Moeneclaey. "Code-aided phase and timing ambiguity resolution for AWGN channels". In *Proceedings IASTED Intl. Conf. on Acoustics, Signal and Image Processing (SIP03)*, Honolulu, Hawaii, August 2003.

[174] H. Wymeersch and M. Moeneclaey. "ML rate detection for multi-rate TH-UWB impulse radio". In *Proc. IEEE Int. Conference on Ultra-Wideband*, Zürich, Switzerland, September 2005.