Iterative Estimation, Equalization and Decoding

A Thesis
Presented to
The Academic Faculty
By

Renato da Rocha Lopes

In Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy in Electrical Engineering

School of Electrical and Computer Engineering

Georgia Institute of Technology

July 8, 2003

Iterative Estimation, Equalization and Decoding

Approved:

_____
John R. Barry, Chairman

_____
Steven W. McLaughlin

_____
Aaron Lanterman

Date Approved _____

To my beloved wife Túria,

with my endless gratitude, love and admiration.

Acknowledgments

First, I would like to thank Dr. John Barry for his guidance during my stay at Georgia Tech. His very clear and objective view of technical matters, as well as his deep insights, have influenced a lot my view of research in telecommunications, and hopefully made this work more clear and objective.

I would also like to thank Drs. Lanterman and McLaughlin for their constructive and timely feedback on my thesis, and Drs. Stüber and Wang for being part of my defense committee.

I am grateful for the help of the School of Electrical and Computer Engineering: Dr. Sayle, Dr. Hertling, Marilou and others were always there when needed. Also, I would like to thank the School of ECE, CRASP and the Brazilian government, through CAPES, for their financial support during different stages of this program.

I am deeply indebted to my wife, Túria, for her support, love and encouragement, and for having pushed me when I needed pushing. I consider myself blessed to be married to such a wonderful person. Without her, this work and a lot more would not have been possible. For all she has done and put up with, I dedicate this thesis to her.

I would also like to thank my parents for their unconditional support and love. They have always taught me the importance of a positive attitude, and how learning can be fun. These lessons, and their unshakable believe in me, have been very important for the completion of my studies.

I am also grateful to my parents-in-law, who have shown great patience and faith.

Thanks are also in order to all the folks at GCATT: Mai, Estuardo, Andrew, Elizabeth, Jau, Apu, Aravind, Badri, Ana, Ravi, Kofi, Chen-Chu, Ricky, Babak, Cagatai, and the list could go on forever. These incredible people showed me how vast the area of telecommunications is, and helped me have a deeper understanding of many research questions. And the discussions with them about music, cooking, religion, politics, cricket, etc., greatly broadened my horizons. They helped make this experience all the more worth it.

Finally, there is the Brazilian crowd of Atlanta, who provided delightful breaks from the daily struggles of the Ph.D. program, and from the occasional struggle of life abroad: Ânderson, Mônica, Pedro, Jackie, Sharlles, Adriane, Henrique, Sônia, Augusto, the musicians, and many others. *Valeu*!

Table of Contents

List of Figures

Summary

Knowledge of the channel is valuable for equalizer design. To estimate the channel, a training sequence, known to the transmitter and the receiver, is normally transmitted. However, transmission of a training sequence decreases the system throughput. Blind channel estimation uses only the statistics of the transmitted signal. Thus, it requires no training sequence, increasing the throughput.

Most real-life communication systems employ some form of error-control code (ECC) to improve the system performance under noise. In fact, with the advent of turbo codes and turbo equalization, reliable transmission at a signal to noise ratio (SNR) close to capacity is now feasible. However, blind estimators that ignore the code may fail at low SNR. Recently, blind estimators have been proposed that exploit the ECC and work well at low SNR. These algorithms are inspired by turbo equalizers and the expectation-maximization (EM) channel estimator.

The objective of this research is to develop a low-complexity ECC-aware blind channel estimator. We first propose the extended-window (EW) algorithm, a channel estimator that is less complex than the EM estimator, and has better convergence properties. Furthermore, the EM algorithm uses the computationally complex forward-backward recursion (BCJR algorithm) for symbol estimation. With the EW estimator, any soft-output equalizer may be used, allowing for further complexity reduction.

We then propose the soft-feedback equalizer (SFE), a low-complexity soft-output equalizer that can use *a priori* information on the transmitted symbols, and is thus suitable for turbo equalization. The coefficients of the SFE are chosen to minimize the mean-squared error between the equalizer output and the transmitted symbols, and depend on the "quality" of the *a priori* information and the equalizer output. Simulation results show that the SFE may perform within 1 dB of a system using a BCJR equalizer, and outperforms other schemes of comparable complexity.

Finally, we show how the SFE and the EW algorithms may be combined to form the turbo estimator (TE), a linear-complexity ECC-aware blind channel estimator. We show that the TE performs close to systems with channel knowledge at low SNR, where ECC-ignorant channel estimators fail.

CHAPTER 1

Introduction

Error-control codes (ECC), or channel codes, allow for reliable transmission of digital information in the presence of noise. Through this process, an information-bearing sequence of length K, called a *message*, is mapped, or encoded, into another sequence of length $N > K$, called a *codeword*. This encoding introduces redundancy, but it also restricts the number of possible transmitted codewords, allowing for reliable communication at a lower signal-to-noise ratio (SNR) [1]. The codeword is then modulated and transmitted through the communications channel. The received signal is a distorted version of the modulated codeword; in particular, communications channels introduce noise, normally modeled as additive white Gaussian noise (AWGN), and intersymbol interference (ISI), the effects of which are normally modeled by a linear filter.

The receiver goal can be very clearly and concisely described: the transmitted message bits should be estimated at the receiver according to a rule that minimizes the bit error rate (BER). Assuming equally likely message bits, this rule can be implemented with a maximum-likelihood (ML) detector, which estimates each message bit so as to maximize the likelihood of observing the received signal conditioned on the message bit [1].

ML detectors jointly and optimally perform all receiver tasks, such as synchronization, timing recovery, channel estimation, equalization, demodulation and decoding. Unfortunately, the computational complexity of ML receivers is prohibitive. In some cases, such as coded systems with interleavers, an ML receiver has to consider every possible transmitted message independently. For messages of 1,000 bits, this means considering $2^{1,000}$ messages, much more than the current estimate for the number of atoms in the universe [2]. Until the promise of quantum computers (which theoretically could analyze all possible messages simultaneously) is realized [3], or until a better strategy is discovered, exact ML detection will remain a benchmark and an object of theoretical investigation.

Traditionally, receivers employ a suboptimal divide-and-conquer approach for recovering the transmitted message from the received signal. First, timing is estimated [1] and the signal is sampled. Then the equalizer parameters are estimated [1,4-8]. After that, the equalizer removes the ISI introduced by the channel [1], so that its output can be seen as a noise-corrupted version of the transmitted codeword. Finally, the equalizer output is fed to the channel decoder which, exploiting the beneficial effects of channel encoding, estimates the transmitted message [2].

The divide-and-conquer approach is clearly suboptimal. Consider, for instance, the problem of channel estimation. Traditionally, the channel is estimated by transmitting a known sequence, called a training sequence [1,4,5], and the received samples corresponding to the training sequence are used for estimation. However, this approach, known as *trained* estimation, ignores received samples corresponding to the information bits, and thus does not use all the information available at the receiver. To improve

performance, the channel may be estimated based on all received samples, in what is known as *semi-blind* estimation [6]. Channel estimation is still possible even if no training sequence is available. In this case, we obtain *blind* channel estimates [7,8]. When performing blind or semi-blind channel estimation under the divide-and-conquer framework, the fact that the transmitted signal is restricted to be a codeword of a given channel code is not exploited. However, it seems clear that performance would be improved if the channel encoding were taken into account.

In this work, we propose the blind turbo estimator (TE), a low computational complexity technique for exploiting the presence of ECC in blind and semi-blind channel estimation. This work has four facets: channel estimation, blind and semi-blind techniques, the exploitation of ECC, and low computational complexity. The importance of each facet is discussed below:

- Channel estimation. Channel estimates are required by the ML equalizer, and can be used to compute the coefficients of suboptimal but lower-complexity equalizers such as the minimum mean-squared error (MMSE) linear equalizer (LE) [1], or the MMSE decision-feedback equalizer (DFE) [1]. Even though the MMSE-LE and the MMSE-DFE can be estimated directly, having the channel estimates allows us to choose which equalizer is more appropriate for the channel. For instance, in channels with deep spectral nulls, DFE is known to perform better than LE.

- Blind and semi-blind techniques. By using every available channel output for channel estimation, semi-blind techniques perform better than techniques based solely on the channel outputs corresponding to training symbols and thus can use a shorter training sequence [6]. Therefore, semi-blind and blind techniques increase

the throughput of a system by requiring a small training sequence or none whatsoever. Eavesdropping is another application of blind channel estimation. The eavesdropper may not know what the training sequence is and hence has to rely on blind estimation techniques.

- Exploitation of ECC. Most of the existing blind channel estimation techniques operate within the divide-and-conquer framework, ignoring the presence of ECC, and normally assuming that the transmitted symbols are independent and identically distributed (iid). This approach works well at high signal-to-noise ratio (SNR). However, the last decade has seen the discovery of powerful ECC techniques such as turbo codes and low-density parity check codes [9-11] that, with reasonable complexity, allow reliable transmission at an SNR only fractions of a dB from channel capacity. When powerful codes are used and systems operate at low SNR, blind and semi-blind estimation techniques that ignore ECC are doomed to fail. This observation motivated the study of blind ECC-aware channel estimators in [12-17].

- Low computational complexity. The per-symbol computational complexity of existing ECC-aware channel estimators is exponential in the memory of the channel. However, in applications such as xDSL and high-density magnetic recording, the channel impulse can have tens or even hundreds of coefficients. For channels with long memory, existing ECC-aware channel estimators are prohibitively complex, which motivates the study of low-complexity techniques.

Examples abound that show that it is possible to improve the performance of the divide-and-conquer approach simply by having the receiver components cooperate through an iterative exchange of information. For instance, in turbo equalizers [18,19] (which assume channel knowledge), the decoder output is used by the equalizer as *a priori* information on the transmitted symbols. This produces improved equalizer outputs, which in turn produce improved decoder outputs, and so on. By iterating between the equalizer and the decoder, turbo equalizers achieve a BER much smaller than that of the divide-and-conquer approach, with reasonable complexity. Iterative channel estimators [6,20-28] are another important class of iterative algorithms that perform better than their noniterative counterparts. In these algorithms, an initial channel estimate is used by a symbol estimator to provide tentative estimates of the first- and/or second-order statistics of the transmitted symbol sequence. These statistics are then used by a channel estimator to improve the channel estimates. The improved channel estimates are then used by the symbol estimator to improve the estimates of the statistics, and so on.

Turbo equalizers and iterative channel estimators normally rely on the forward-backward algorithm by Bahl, Cocke, Jelinek and Raviv (BCJR) [29] for equalization. This algorithm computes the *a posteriori* probabilities (APP) of the channel inputs given the channel output, channel estimates, and *a priori* probabilities on the channel inputs, and assuming that the channel inputs are independent. In other words, if an ECC is present, this presence is ignored.

The BCJR algorithm is well-suited for iterative systems, since it can use the *a priori* information at its input to improve the quality of its output and since it computes soft symbol estimates in the form of APP. However, its per-symbol computational complexity

increases exponentially with the channel memory, and hence is prohibitive for channels with a long impulse response. This has motivated the development of reduced-complexity alternatives to the BCJR algorithm, such as the equalizers proposed in [30-38]. The structures proposed in [30-35] use a linear filter to equalize the received sequence. The output of this filter contains residual ISI, which is estimated based on the *a priori* information, and then cancelled. Reduced-state algorithms are investigated in [36-38]; however, these are more complex than the structures based on linear filters.

In this work, we propose the *soft-feedback equalizer* (SFE), a low-complexity alternative to the BCJR algorithm based on filters that is similar to those proposed in [30-35]. One important difference is that the SFE uses a structure similar to a DFE, combining the equalizer outputs and *a priori* information to form more reliable estimates of the residual ISI. A similar system is proposed in [35] that uses hard decisions on the equalizer output to estimate the residual ISI. However, because hard decisions are used and because the equalizer output is not combined with the *a priori* information before a decision is made, the DFE-like system of [35] performs worse than schemes without feedback.

As in [32-35], the SFE does not rely solely on interference cancellation (IC). Instead, the SFE coefficients are computed so as to minimize the mean-squared error (MSE) between the equalizer output and the transmitted symbol. The resulting equalizer coefficients depend on the quality of the equalizer output and the *a priori* information. By assuming a statistical model for the equalizer outputs and the *a priori* information, we obtain a linear-complexity, time-invariant equalizer. In contrast, the MMSE structures in [32-35] have to be computed for every symbol, resulting in a per-symbol complexity that

is quadratic in the length of the equalizer. A similar statistical model is used in [39] to obtain a time-invariant, linear-complexity, hard-input hard-output equalizer with ISI cancellation.

We will see that in special cases, the SFE reduces to an MMSE-LE, an MMSE-DFE, or an IC. We will show that the SFE performs reasonably well when compared to the BCJR algorithm and the quadratic complexity algorithms in [32-35], while it outperforms other structures of comparable complexity proposed in the literature.

Iterative channel estimators fit the iterative paradigm depicted in Fig. 1. In this figure, a symbol estimator produces soft information on the transmitted symbols based on the channel estimates, $\hat{\boldsymbol{h}}$, and the noise variance estimate, $\hat{\sigma}^2$, provided by the channel estimator. The channel estimator then uses the soft information on the transmitted symbols to compute improved channel estimates. The new channel estimates are then used by the symbol estimator to compute better soft information, and so on. The most important iterative estimator, on which most other iterative estimators are based, is the expectation-maximization (EM) algorithm [40,41]. In this algorithm, the symbol estimator in Fig. 1 is based on the BCJR algorithm and produces soft estimates of the first- and second-order statistics of the transmitted symbols.

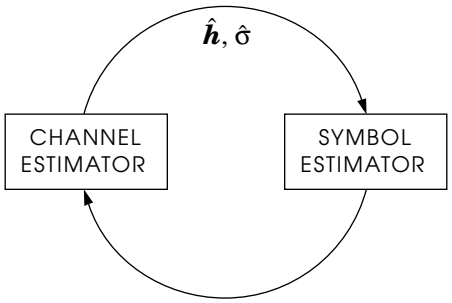

Fig. 1. Blind iterative channel estimation.

The EM algorithm has two sources of complexity. First, it involves the computation and inversion of a square matrix whose order is equal to the channel length. Second, and most important, it uses the BCJR algorithm for equalization. In this work, we will obtain a *simplified EM* (SEM) algorithm that avoids the matrix inversion without significantly affecting performance, resulting in a complexity that is proportional to the channel length. More interestingly, based on the SEM, the soft symbol estimator may be implemented with any of a number of low-complexity alternatives to the BCJR algorithm, such as the SFE. Low complexity alternatives to the EM channel estimator are also proposed in [25,26]. However, in these strategies the complexity is reduced through the use of a low-complexity alternative to the BCJR algorithm. Therefore, they are intrinsically tied to an equalization scheme. Furthermore, the estimators proposed in [25,26] do not avoid the matrix inversion, resulting in a quadratic computational complexity.

In this work, we will also investigate convergence issues regarding iterative channel estimators. The EM algorithm generates a sequence of estimates with nondecreasing likelihood. Hence, the EM estimates may converge to the ML solution. However, they may also get trapped in a nonglobal local maximum of the likelihood function. We will propose a simple modification, called the *extended-window EM* (EW) algorithm, which greatly decreases the probability of misconvergence without significantly increasing the computational complexity.

Finally, by viewing a turbo equalizer as a soft symbol estimator, we combine turbo equalization with iterative channel estimation. Since turbo equalizers provide soft symbol estimates that benefit from the presence of channel coding, the resulting turbo estimation scheme is an ECC-aware channel estimator. Thus, we have proposed the *turbo estimator*

(TE), a linear complexity blind channel estimator that benefits from the presence of channel coding. Other ECC-aware channel estimators were proposed [12-16], but they are all based on the EM algorithm and hence suffer all the complexity and convergence problems mentioned above.

To summarize, the main contributions of this work are:

- The soft-feedback equalizer (SFE), a linear complexity equalizer that produces soft symbol estimates and benefits from *a priori* information at its input.
- The simplified EM (SEM) algorithm, an iterative channel estimator that is less complex than the EM algorithm and is not intrinsically tied to the BCJR equalizer, which opens the door for further complexity reduction.
- The extended window (EW) algorithm, an iterative channel estimator that is less prone to misconvergence than the EM algorithm.
- The turbo estimator (TE), an iterative channel estimator that benefits from the presence of ECC to produce reliable channel estimates at low SNR.

This thesis is organized as follows. In Chapter 2, we present the channel model and describe the problem we will investigate, and provide some background material on turbo equalization and iterative channel estimation via the EM algorithm. In Chapter 3, we propose the SEM, a channel estimator that is less complex than the EM algorithm. In Chapter 4, we propose the EW algorithm, an extension to the SEM algorithm that makes it is less likely than the EM to get trapped in a local maximum of the joint likelihood function. In Chapter 5, we propose the SFE, a linear-complexity alternative to the BCJR equalizer. In Chapter 6, we describe the application of the SFE to turbo equalization.

Please note that Chapters 5 and 6 are not related to Chapters 3 and 4. In Chapter 7, we propose the TE, a linear complexity ECC-aware channel estimator that combines the EW algorithm of Chapter 4 with the SFE-based turbo equalizer of Chapter 6. In Chapter 8 we summarize the contributions of this thesis and present directions for future work.

CHAPTER 2

Problem Statement and Background

In this chapter, we describe the model we will use for the communications system and define the problem investigated in this research. We also provide background material on two iterative techniques that solve parts of this problem: turbo equalization and iterative channel estimation.

2.1 Problem Statement

We consider the system model shown in Fig. 2, where a binary message $\boldsymbol{m} = [m_0, \ldots m_{K-1}]$ of length K is transmitted across a linear AWGN channel with memory μ. The channel and the noise are assumed to be real. A binary ECC encoder with rate K/N maps $\boldsymbol{m}$ to a sequence of binary phase-shift keying (BPSK) symbols $\boldsymbol{c} = [c_0, \ldots c_{N-1}]$ of length N. As with wireless systems and systems employing turbo equalization, the codeword $\boldsymbol{c}$ is

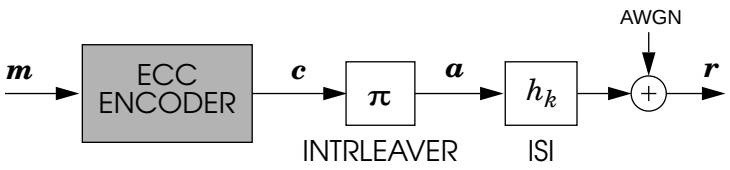

Fig. 2. Channel model.

permuted according to the interleaver $\boldsymbol{\pi}$ before transmission. Let $\{\pi(0), \ldots \pi(N-1)\}$ be a permutation of $\{0, \ldots N-1\}$. Then, the interleaver output is $\boldsymbol{a} = [a_0, \ldots a_{N-1}]$, with $a_k = c_{\pi(k)}$.

Let $\boldsymbol{r} = [r_0, \ldots r_{L-1}]$ denote the received sequence of length $L = N + \mu$, where

$$r_k = \boldsymbol{h}^{\mathrm{T}} \boldsymbol{a}_k + n_k, \tag{1}$$

where the channel impulse response is $\boldsymbol{h} = [h_0, \ldots h_\mu]^{\mathrm{T}}$, where $\boldsymbol{a}_k = [a_k, \ldots a_{k-\mu}]^{\mathrm{T}}$ is the channel input, and where n_k represents additive white Gaussian noise (AWGN) with variance σ^2. For notational ease, we restrict our presentation to the BPSK alphabet, where $a_k \in \{\pm 1\}$. The results in this work can be extended to other alphabets using the techniques described in [33].

Ideally, we would like to solve the joint-ML blind channel estimation and symbol detection problem, *i.e.*, find

$$(\hat{\boldsymbol{h}}_{\mathrm{ML}}, \hat{\sigma}_{\mathrm{ML}}, \hat{\boldsymbol{m}}_{\mathrm{ML}}) = \operatorname{argmax} \log p_{\boldsymbol{h},\sigma}(\boldsymbol{r} \mid \boldsymbol{m}), \tag{2}$$

where $\log p_{\boldsymbol{h},\sigma}(\boldsymbol{r} \mid \boldsymbol{m})$ is the log-likelihood function, defined as the logarithm of the probability density function (pdf) of the received signal $\boldsymbol{r}$ conditioned on the channel input $\boldsymbol{m}$ and parametrized by $\boldsymbol{h}$ and σ. Intuitively, the ML estimates $\hat{\boldsymbol{h}}_{\mathrm{ML}}$, $\hat{\sigma}_{\mathrm{ML}}$, and $\hat{\boldsymbol{m}}_{\mathrm{ML}}$ are those that best explain the received sequence, in the sense that we are less likely to observe the channel output if we assume any other set of parameters to be correct, *i.e.*, $p_{\boldsymbol{h},\sigma}(\boldsymbol{r} \mid \boldsymbol{m}) \leq p_{\hat{h}_{\mathrm{ML}}, \hat{\sigma}_{\mathrm{ML}}}(\boldsymbol{r} \mid \hat{\boldsymbol{m}}_{\mathrm{ML}}) \ \forall \ \boldsymbol{h}, \sigma, \boldsymbol{m}$. Besides this intuitive interpretation, ML estimates have many interesting theoretical properties [4]. Under fairly general conditions, ML estimates are asymptotically unbiased and efficient. In other words, under these general conditions and as the number of transmitted symbols N tends to infinity, the expected value of the ML

estimates tends to the actual value of the parameters, while the variance of the estimates tends to the Cramér-Rao bound, which is the lowest variance achievable by any unbiased estimator.

Unfortunately, the computational complexity of finding the ML estimates is prohibitive. In this work, we will study iterative approaches that provide approximate solutions to the maximization problem in (2). The reason for the focus on iterative approaches is that iterative techniques successfully provide approximate ML solutions to otherwise intractable problems, such as the following:

- On a coded system with channel knowledge, turbo equalizers produce a good approximation, with reasonable computational complexity, to the maximization of $\log p(\boldsymbol{r} \mid \boldsymbol{m})$.
- On an uncoded system, the EM algorithm provides a simple approximate ML channel estimate for the blind ECC-ignorant problem of maximizing $\log p_{\boldsymbol{h},\sigma}(\boldsymbol{r} \mid \boldsymbol{a})$. Here, $\boldsymbol{a}$ is not restricted to be a permutation of a codeword, but instead can be any vector of symbols of length N.

These techniques are formulated in a framework that makes it almost straightforward to combine them in a more general iterative algorithm that performs channel identification and decoding, as we will see in chapter 7.

One key ingredient of a successful iterative algorithm is the use of soft symbol estimates in the form of APP's. For a general alphabet $\mathcal{A}$, the APP is a function from $\mathcal{A}$ to the interval [0,1] given by $\Pr(a_k = a \mid \boldsymbol{r})$, for $a \in \mathcal{A}$. For a BPSK constellation, the APP is fully captured by what is loosely referred to as the log-likelihood ratio (LLR), defined as

$$L_k = \log \frac{\Pr(a_k = +1 | \boldsymbol{r})}{\Pr(a_k = -1 | \boldsymbol{r})}. \tag{3}$$

The LLR has some interesting properties. For a BPSK alphabet, the sign of L_k determines the maximum *a posteriori* (MAP) estimate of a_k, which minimizes the probability of a decision error, and its magnitude provides a measure of the reliability of the decision. Furthermore, L_k can be used to obtain the MMSE estimate of a_k, which, for a BPSK alphabet, is given by $\tanh(L_k / 2)$.

Unfortunately, exact evaluation of the APP is computationally hard. In the next two sections, we will briefly review turbo equalizers and the EM algorithm, which are iterative techniques that address simpler problems and are the building blocks for the system proposed in this work.

2.2 Turbo Equalization

Assuming channel knowledge, the goal of the decoder is to estimate $\Pr(m_k = 1 | \boldsymbol{r})$ for each message bit m_k, which is a computationally hard problem. Turbo equalizers, first proposed in [18], provide a low complexity approximate solution to this problem. In this section, we review the turbo equalization algorithm.

Turbo equalizers consist of one soft-input soft-output (SISO) equalizer, one interleaver π, one deinterleaver π^{-1}, and one SISO channel decoder, as shown in Fig. 3 for a BPSK

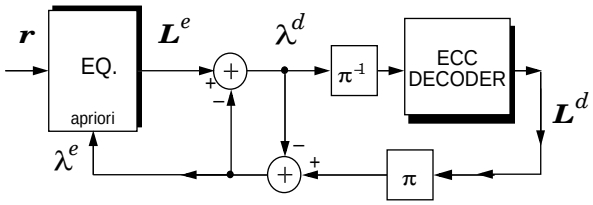

Fig. 3. Turbo equalizer.

alphabet. Key to the low complexity of turbo equalizers is the fact that the SISO equalizer ignores the presence of ECC, and the SISO decoder ignores the presence of the channel. The resulting complexity is thus of the same order of magnitude as that of the divide-and-conquer approach employing the same equalizer and decoder.

Turbo equalization is an iterative, block-processing algorithm whose first iteration is the same as a divide-and-conquer detector. Indeed, the vector of *a priori* information at the equalizer input, $\boldsymbol{\lambda}^e = [\lambda_0^e, \ldots \lambda_{N-1}^e]$, is initially set to zero. The SISO equalizer then computes the LLR vector $\boldsymbol{L}^e = [L_0^e, \ldots L_{N-1}^e]$ of the codeword symbols a_k given the channel observations $\boldsymbol{r}$. These LLRs are computed exploiting only the structure of the ISI channel; the ECC encoder is ignored. The equalizer output is then deinterleaved by the deinterleaver $\boldsymbol{\pi}^{-1}$ and passed to the decoder. Finally, using the deinterleaved values of $\boldsymbol{L}^e$ and exploiting the code structure (the ISI channel is ignored, presumably because the equalizer has removed its effects), the SISO decoder computes new LLRs of each codeword symbol, $\boldsymbol{L}^d = [L_0^d, \ldots L_{N-1}^d]$.

The difference between the first iteration and the later ones is that, for later iterations, information is fed back from the decoder to the equalizer through $\boldsymbol{\lambda}^e$, which is used as *a priori* information by the equalizer. This feedback of information allows the equalizer to benefit from the code structure, which provides improved soft information at the decoder output. However, $\boldsymbol{\lambda}^e$ does not correspond to the full probabilities at the decoder output. Instead, as seen in Fig. 3, it is the difference between the LLRs at the input and the output of the decoder. With this subtraction, λ_k^e is not a function of L_k^e, avoiding positive feedback of information back to the equalizer. The LLRs in $\boldsymbol{\lambda}^e$ are called *extrinsic information* and

can be seen as the information on the transmitted symbols gleaned by exploiting only the structure of the decoder. The extrinsic information at the decoder input, λ^d, can be similarly defined.

2.2.1 The BCJR Algorithm

Ideally, the equalizer in Fig. 3 should be implemented with the BCJR algorithm, which computes the APPs of the transmitted symbols given their *a priori* probabilities and the channel observations. (Note that the only way ECC affects the BCJR equalizer is through the *a priori* information; it is otherwise ignored.) Actually, the BCJR algorithm can be defined for any trellis, and hence can also be used to implement the decoder for a convolutional code. In the sequel, we will describe the BCJR algorithm for equalization in detail and then discuss the differences between the BCJR equalizer and the BCJR decoder.

Let $\psi_k \in \{0, 1, \ldots Q-1\}$ denote a state of the channel trellis at time k, where $Q = |\mathcal{A}|^{\mu}$ is the number of states and $|\mathcal{A}|$ is the number of elements in the alphabet. Note that there is a one-to-one correspondence between the value of ψ_k and the vector of symbols in the channel memory, $[a_{k-1} \ldots a_{k-\mu}]$. Also, let $a^{(p,q)}$ be the channel input that causes the transition from state p to state q. Then, the APP $\Pr(a_k = a \mid \boldsymbol{r})$ can be computed as [29]

$$\Pr(a_k = a \mid \boldsymbol{r}) = \sum_{\{p,\, q : a^{(p,q)} = a\}} \Pr(\psi_k = p;\, \psi_{k+1} = q \mid \boldsymbol{r}). \tag{4}$$

The key observation leading to the BCJR algorithm is that the terms in the summation in (4) can be decomposed into three factors, one depending only on past channel outputs grouped in the vector $\boldsymbol{r}_{l<k}$, one depending only on future channel outputs grouped in the vector $\boldsymbol{r}_{l>k}$, and one depending on the current channel output r_k. Indeed, exploiting the fact that the trellis corresponds to a Markov process, we get, after some manipulation [29],

$$\Pr(\psi_k = p;\, \psi_{k+1} = q \mid \boldsymbol{r}) = \alpha_k(p)\, \gamma_k(p,q)\, \beta_{k+1}(q) / p(\boldsymbol{r}), \tag{5}$$

where

$$\alpha_k(p) = p(\psi_k = p;\, \boldsymbol{r}_{l<k}), \tag{6}$$

$$\beta_{k+1}(q) = p(\boldsymbol{r}_{l>k} \mid \psi_{k+1} = q), \tag{7}$$

$$\gamma_k(p,q) = p(\psi_{k+1} = q;\, r_k \mid \psi_k = p). \tag{8}$$

Note that, since we are interested in probability ratios, the factor $p(\boldsymbol{r})$ in equation (5) is irrelevant.

After further manipulation and consideration of the Markov property, the following recursions can be found for computing $\alpha_k(p)$ and $\beta_k(p)$ [29]:

$$\alpha_k(p) = \sum_{q=0}^{Q-1} \alpha_{k-1}(q)\, \gamma_k(q,p) \tag{9}$$

$$\beta_k(p) = \sum_{q=0}^{Q-1} \beta_{k+1}(q)\, \gamma_k(p,q). \tag{10}$$

The recursions in (9) and (10) can lead to underflow on finite precision computers. To avoid this problem, it is common to normalize α_k and β_k at each time k, so that $\sum_p \alpha_k(p) = 1$ and $\sum_p \beta_k(p) = 1$ [42].

Finally, to compute $\gamma_k(p, q)$, write

$$\gamma_k(p, q) = p(\psi_{k+1} = q; r_k \mid \psi_k = p) \tag{11}$$

$$= p(r_k \mid \psi_k = p; \psi_{k+1} = q)\Pr(\psi_{k+1} = q \mid \psi_k = p). \tag{12}$$

The second term in equation (12) is the probability that the channel input is the one that causes a transition from state p to state q, $a^{(p,q)}$. Thus, this term is the *a priori* information of the input of the SISO block. For a BPSK alphabet, when this *a priori* information is given in the form of the LLR λ_k^e, we get

$$\Pr(\psi_{k+1} = q \mid \psi_k = p) = \frac{e^{a^{(p,q)}\lambda_k^e/2}}{e^{\lambda_k^e/2} + e^{-\lambda_k^e/2}}. \tag{13}$$

Note that the denominator in (13) is common to all state transitions. Thus, since we are interested in computing probability ratios, the denominator in (13) may be ignored.

Assuming AWGN, the first term of equation (12) can be computed as

$$p(r_k \mid \psi_k = p; \psi_{k+1} = q) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2} \mid r_k - r^{(p,q)} \mid^2\right\}, \tag{14}$$

where $r^{(p,q)}$ is the noiseless channel output associated with the transition from state p to state q. This completes the description of the BCJR algorithm for equalization.

Since convolutional codes may also be defined by a trellis, the BCJR algorithm may also be used for decoding these codes. The algorithms for decoding and equalization proceed in a similar manner. The main differences are as follows:

- The equalizer computes only the APPs of the channel inputs. In contrast, the decoder computes the APP of the encoder output, a_k, as well as the encoder input m_k, which will provide a MAP estimate of the transmitted message. Both these

APPs may be computed by considering the appropriate state transitions in the summation in (4).

- While for the equalizer each trellis stage corresponds to a single channel output, for the decoder a trellis stage may correspond to multiple outputs. For instance, a rate 1/2 convolutional code has two outputs for every state transition. In general, for a rate k/n convolutional code, $|r_k - r^{(p,q)}|$ in (14) is the distance between vectors of length n.

- For turbo equalizers, such as the structure depicted in Fig. 3, the decoder does not have access to the channel output. Therefore, (14) reduces to a constant, and the state transition probability $\gamma_k(p, q)$ depends only on the *a priori* information.

2.3 Blind Iterative Channel Estimation with the EM Algorithm

In many ML estimation problems, the difficulty in finding a solution stems from the fact that some information about how the observed data was generated is missing. For instance, in the blind channel estimation problem, finding the ML channel estimates would be easy if the channel inputs were known. For ML problems that would be easily solvable if the missing data were available, the EM algorithm is an interesting approach. It is a low-complexity iterative algorithm that generates a sequence of estimates with non-decreasing likelihood. Thus, with proper initialization, or if the likelihood function does not posses local maxima, the EM algorithm will converge to the ML solution. In the remainder of this section, we will describe the EM algorithm for blind channel estimation, as first proposed in [22].

In its most general form [40,41], the EM algorithm can be described as follows. Let $p_\theta(\boldsymbol{r})$ be the likelihood function of the received samples, where θ is the vector of parameters we are trying to estimate. Let $\boldsymbol{a}$ be the missing data, and $\boldsymbol{r}$ be the sequence of observations. Assume we have an estimate θ_i of the parameters. Define the auxiliary function

$$Q(\theta, \theta_i) = \int \log(p_\theta(\boldsymbol{r}, \boldsymbol{a})) p_{\theta_i}(\boldsymbol{a} \mid \boldsymbol{r}) \, d\boldsymbol{a} = E_{\boldsymbol{a}, \theta_i}[\log(p_\theta(\boldsymbol{r}, \boldsymbol{a})) \mid \boldsymbol{a}], \tag{15}$$

where $E_{\boldsymbol{a}, \theta_i}$ is the expected value with respect to the variable $\boldsymbol{a}$, assuming that the actual parameters are θ_i. Now, consider computing a new estimate θ_{i+1} of the parameters according to

$$\theta_{i+1} = \text{argmax}(Q(\theta, \theta_i)). \tag{16}$$

The key observations leading to the EM algorithm are that

- with an appropriate choice of $\boldsymbol{a}$, computing and maximizing $Q(\theta, \theta_i)$ may be an easy task.
- the likelihood of the new estimate θ_{i+1} is not smaller than that of θ_i, *i.e.*, $p_{\theta_{i+1}}(\boldsymbol{r}) \geq p_{\theta_i}(\boldsymbol{r})$.

Given an initial estimate θ_0, the EM algorithm iteratively computes new estimates using (15) and (16) until a stop criterion is met, thus generating a sequence of estimates with nondecreasing likelihood.

When applied to the problem of blind channel estimation, the EM algorithm may be described in more specific terms. In this case, we are trying to maximize the likelihood function $p_\theta(\boldsymbol{r}) = \log p_{\boldsymbol{h},\sigma}(\boldsymbol{r})$, where $\theta = [\boldsymbol{h}, \sigma]$ is the vector of parameters we are trying to

estimate. As mentioned before, finding the values of $\boldsymbol{h}$ and σ that maximize this likelihood function is prohibitively complex. However, it is easy to determine the parameters that maximize $p_\theta(\boldsymbol{r} \mid \boldsymbol{a})$, in which case the solution is a simple MMSE channel estimate [4]:

$$\hat{\boldsymbol{h}} = \left(\sum_{k=0}^{N-1} \boldsymbol{a}_k \boldsymbol{a}_k^T \right)^{-1} \sum_{k=0}^{N-1} r_k \boldsymbol{a}_k , \tag{17}$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{k=0}^{N-1} \left(r_k - \hat{\boldsymbol{h}}^T \boldsymbol{a}_k \right)^2 . \tag{18}$$

Thus, the transmitted symbols $\boldsymbol{a}$ are a good candidate for hidden information.

Having defined the missing variables, we can apply (15) and (16) to compute new channel estimates. Let

$$\hat{\mathbf{R}}_{\boldsymbol{a}} = \frac{1}{N} \sum_{k=0}^{N-1} \mathrm{E}[\boldsymbol{a}_k \boldsymbol{a}_k^T \big| \boldsymbol{r}] \tag{19}$$

$$\hat{\boldsymbol{p}}_{\boldsymbol{a}r} = \frac{1}{N} \sum_{k=0}^{N-1} r_k \mathrm{E}[\boldsymbol{a}_k \big| \boldsymbol{r}] . \tag{20}$$

Then, it is possible to show that the EM algorithm yields [22]

$$\hat{\boldsymbol{h}}_{i+1} = \hat{\mathbf{R}}_{\boldsymbol{a}}^{-1} \hat{\boldsymbol{p}}_{\boldsymbol{a}r} , \tag{21}$$

$$\begin{aligned} \hat{\sigma}_{i+1}^2 &= \frac{1}{N} \sum_{k=0}^{N-1} \mathrm{E}\left[\left| r_k - \boldsymbol{h}_{i+1}^T \boldsymbol{a}_k \right|^2 \Big| \boldsymbol{r} \right] \\ &= \frac{1}{N} \sum_{k=0}^{N-1} |r_k|^2 - 2 \hat{\boldsymbol{h}}_{i+1}^T \hat{\boldsymbol{p}}_{\boldsymbol{a}r} + \hat{\boldsymbol{h}}_{i+1}^T \hat{\mathbf{R}}_{\boldsymbol{a}} \hat{\boldsymbol{h}}_{i+1} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} |r_k|^2 - \hat{\boldsymbol{h}}_{i+1}^T \hat{\boldsymbol{p}}_{\boldsymbol{a}r} , \end{aligned} \tag{22}$$

where the last equality follows from (21).

Note the similarities between (17) and (21), and between (18) and (22). The only difference between these equations is that in (17) and (18) the actual transmitted sequence is used, while in (21) and (22) the conditional *a posteriori* expected values are used. In

fact, $\hat{\mathbf{R}}_{\boldsymbol{a}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{a}r}$ are similar to the estimated autocorrelation matrix of $\boldsymbol{a}$ and the estimated cross-correlation vector between $\boldsymbol{a}$ and $\boldsymbol{r}$, respectively. The main difference is that we use $\mathrm{E}[\boldsymbol{a}_k \boldsymbol{a}_k^T | \boldsymbol{r}]$ and $r_k \mathrm{E}[\boldsymbol{a}_k | \boldsymbol{r}]$ to compute $\hat{\mathbf{R}}_{\boldsymbol{a}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{a}r}$, while $\boldsymbol{a}_k \boldsymbol{a}_k^T$ and $r_k \boldsymbol{a}_k$ are used to estimate the autocorrelation matrix of $\boldsymbol{a}$ and the cross-correlation vector between $\boldsymbol{a}$ and $\boldsymbol{r}$. Thus, we say that $\hat{\mathbf{R}}_{\boldsymbol{a}}$ is an *a posteriori* sample autocorrelation matrix and $\hat{\boldsymbol{p}}_{\boldsymbol{a}r}$ is an *a posteriori* sample cross-correlation vector.

We still have to compute the values of $\mathrm{E}[\boldsymbol{a}_k \boldsymbol{a}_k^T | \boldsymbol{r}]$ and $\mathrm{E}[\boldsymbol{a}_k | \boldsymbol{r}]$ at every iteration i. This can be done with the BCJR algorithm, which is used under the assumption that the channel parameters are given by $\hat{\boldsymbol{h}}_i$ and $\hat{\sigma}_i$. Since the BCJR algorithm computes $\Pr(a_k | \boldsymbol{r})$, obtaining $\mathrm{E}[\boldsymbol{a}_k | \boldsymbol{r}]$ is straightforward. Also, note that each state transition in the channel trellis actually corresponds to a vector $\boldsymbol{a}_k$. Thus, since the BCJR algorithm computes the probabilities of state transition, $\Pr[\psi_k = p; \psi_{k+1} = q | \boldsymbol{r}]$, we in fact have access to the joint APP of the vector $\boldsymbol{a}_k$, which can be used to compute $\mathrm{E}[\boldsymbol{a}_k \boldsymbol{a}_k^T | \boldsymbol{r}]$.

The EM algorithm for blind channel estimation is summarized in the following pseudocode:

```
Given: initial channel estimates ĥ_0 and σ̂_0^2.
i = 0;
repeat
     run the BCJR algorithm, assuming the channel is given by
          ĥ_i and σ̂_i^2;
     compute R̂_a and p̂_ar as in (19) and (20);
     compute the new parameter estimates as in (21) and (22);
until a stop criterion is found
```

This pseudocode can be represented graphically as in Fig. 4. In this figure, the BCJR algorithm is used to compute $\hat{\mathbf{R}}_{\boldsymbol{a}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{a}r}$ based on the channel estimates provided by the channel estimator. The channel estimator then uses the outputs of the BCJR algorithm to compute new channel estimates, which are then used by the BCJR algorithm, and so on.

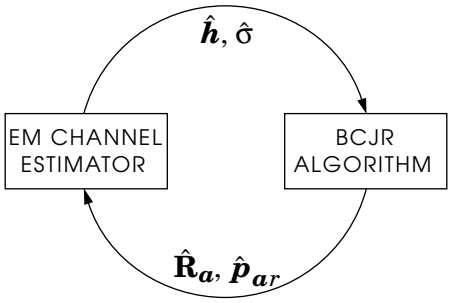

Fig. 4. The EM algorithm for blind iterative channel estimation.

The main drawbacks of the EM algorithm are that

- it uses the BCJR algorithm to produce tentative symbol estimates, implying a computational complexity that is exponential in the channel memory;
- it requires the computation of $\hat{\mathbf{R}}_{\boldsymbol{a}}$ and the solution of the linear system in (21), which have a computational complexity that is quadratic in the channel memory;
- it may get trapped in a local maximum of the likelihood function, converging to wrong channel estimates;
- it may converge slowly.

In the following chapters, we propose techniques to circumvent these drawbacks. We will propose a linear complexity technique that avoids some of the local maxima of the likelihood function that trap the EM algorithm.

CHAPTER 3

A Simplified EM Algorithm

As mentioned in Chapter 2, some of the complexity issues associated with the EM algorithm stem from the need to compute and invert the *a posteriori* sample autocorrelation matrix $\hat{\mathbf{R}}_{\boldsymbol{a}}$ defined in (19). In this chapter, we derive the simplified EM algorithm (SEM), an alternative iterative channel estimator that ignores $\hat{\mathbf{R}}_{\boldsymbol{a}}$ and yet does not significantly degrade the performance relative to the EM algorithm. For notational convenience, in what follows we assume that the transmitted symbols belong to a BPSK constellation. Generalization to other constellations is straightforward.

3.1 Derivation of the SEM Algorithm

Consider the channel model in equation (1), repeated here for convenience

$$r_k = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{a}_k + n_k. \tag{23}$$

Assuming that the transmitted symbols are uncorrelated, we have

$$h_n = \mathrm{E}[r_k a_{k-n}] \tag{24}$$

$$= \mathrm{E}[r_k \Pr(a_{k-n} = +1 \mid r_k)] - \mathrm{E}[r_k \Pr(a_{k-n} = -1 \mid r_k)] \tag{25}$$

$$= \mathrm{E}[r_k \Pr(a_{k-n} = +1 \mid \boldsymbol{r}) - r_k \Pr(a_{k-n} = -1 \mid \boldsymbol{r})] \tag{26}$$

$$= \mathrm{E}[r_k \, \mathrm{E}[a_{k-n} \mid \boldsymbol{r}]]. \tag{27}$$

This equation leads to a simple channel estimator. Unfortunately, the channel estimator has no access to $\mathrm{E}[a_k \mid \boldsymbol{r}]$, which requires exact channel knowledge. However, based on the iterative paradigm of Fig. 1, at the i-th iteration the channel estimator does have access to $\tilde{a}_k^{(i)} = \mathrm{E}[a_k | \boldsymbol{r}; \hat{\boldsymbol{h}}_i, \hat{\sigma}_i] = \tanh(L_k/2)$. Using this value in (27), and also replacing ensemble average with time average, the channel estimate at the i+1-st iteration is given by:

$$\hat{h}_{n,\,i+1} = \frac{1}{N}\sum_{k=0}^{N-1} r_k \tanh\left(\frac{L_{k-n}}{2}\right). \tag{28}$$

Thus, (28) provides a method for estimating the channel given the soft symbol estimates L_k, and can be used in the same context as the channel estimation step of the EM algorithm. Clearly, its implementation has a per-symbol complexity that is linear in the length of the channel if the LLRs are given.

For estimating the noise variance σ^2 at the i-th iteration, we propose using the channel estimates in (28) and the bit estimates obtained from L_k to estimate the noise component of the received signal, which are then used to estimate the noise variance. In other words, letting $\hat{\boldsymbol{a}}_k = [\hat{a}_k, \ldots \hat{a}_{k-\mu}]^{\mathrm{T}}$, where $\hat{a}_k = \mathrm{sign}(L_k)$, we estimate the noise variance as

$$\hat{\sigma}_{i+1}^2 = \frac{1}{N}\sum_{k=0}^{N-1} \left| r_k - \hat{\boldsymbol{a}}_k^T \hat{\boldsymbol{h}}_{i+1} \right|^2, \tag{29}$$

where $\hat{\boldsymbol{h}}_{i+1} = [\hat{h}_{0,\,i+1}, \ldots \hat{h}_{\mu,\,i+1}]^{\mathrm{T}}$. This estimate differs from the EM estimate in (22), but in our simulations we noted that using $\hat{a}_k$ instead of $\mathrm{E}[a_k \mid \boldsymbol{r}]$ for estimating the noise variance improved convergence speed. Further justification for the use of hard decisions in (29) will be given in the next section.

The resulting algorithm is described by the following pseudocode:

```
initialize channel estimates ĥ_0 and σ̂_0^2;
i = 0;
repeat
      use channel estimates to compute symbol estimates L_k, for
            k = 0, … N-1;
      update channel estimates using (28) and (29);
      i = i + 1;
until a stop criterion is met
```

This algorithm will be referred to as simplified EM (SEM). Indeed, using the notation from chapter 2, comparing (20) and (28) we see that $\hat{\boldsymbol{h}}_{i+1} = \hat{\boldsymbol{p}}_{\boldsymbol{a}r}$. Therefore, (28) can be seen as a simplification of the EM algorithm wherein $\hat{\mathbf{R}}_{\boldsymbol{a}}$ is replaced by $\mathbf{I}$. It is important to point out that, from (19), $\hat{\mathbf{R}}_{\boldsymbol{a}} \approx \mathbf{I}$ is a reasonable approximation. In fact, $\mathrm{E}[\boldsymbol{a}_k \boldsymbol{a}_k^T | \boldsymbol{r}]$ is the MMSE estimate of $\boldsymbol{a}_k \boldsymbol{a}_k^T$ given the current channel estimate. Thus, these two values are expected to be approximately the same, so that $\hat{\mathbf{R}}_{\boldsymbol{a}}$ is approximately a time-average estimate of the autocorrelation matrix of the transmitted symbols. Since we assumed that the channel input is white, based on the law of large numbers $\hat{\mathbf{R}}_{\boldsymbol{a}}$ should be close to the identity for large enough N.

An important implication of ignoring the matrix $\hat{\mathbf{R}}_{\boldsymbol{a}}$ is that the channel estimator requires only the soft symbol estimates L_k. Thus, we may represent the simplified channel estimator as in Fig. 5, where the symbol estimator is not restricted to be the BCJR

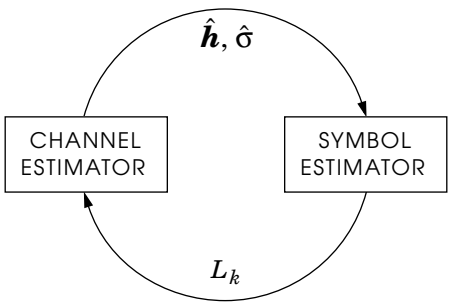

Fig. 5. Blind iterative channel estimation with the SEM algorithm.

equalizer. In fact, any equalizer that produces soft symbol estimates can be used, which allows for a low-complexity implementation of the blind iterative channel estimator. Contrast this figure with Fig. 4, which represents the EM algorithm. In the EM algorithm, the equalizer is restricted to be the BCJR algorithm, and it also must provide a matrix to the channel estimator.

3.2 Analysis of the Scalar Channel Estimator

In this section, we provide a detailed analysis of the SEM algorithm applied to a scalar channel. Although a detailed analysis of the SEM algorithm for a general channel would be of more interest, this analysis is difficult. Furthermore, scalar channels estimators are important, being used in systems that are subject to flat fading [43] and in systems that employ multicarrier modulation [44]. In performing this analysis, we will also compare the performance of systems using soft and hard decisions. In particular, we will justify the use of hard decisions for estimating the noise variance in (29).

Consider the transmission of a sequence of uncorrelated bits $a_k \in \{-1,+1\}$ through a scalar channel with gain A, the output of which is corrupted by an AWGN component n_k with variance σ^2. The received signal can be written as

$$r_k = A\, a_k + n_k. \tag{30}$$

Given initial estimates $\hat{A}_0$ and $\hat{\sigma}_0$, the channel gain and noise variance can be estimated with an iterative algorithm. Possible estimators can be expressed as

$$\hat{A}_{i+1} = \frac{1}{N}\sum_{k=0}^{N-1} r_k \mathrm{dec}_{\mathrm{A}}\left(\frac{1}{2}\hat{L}_i r_k\right), \tag{31}$$

$$\hat{\sigma}_{i+1}^2 = \frac{1}{N}\sum_{k=0}^{N-1}\left| r_k - \hat{A}_{i+1}\mathrm{dec}_\sigma\left(\frac{1}{2}\hat{L}_i r_k\right)\right|^2, \tag{32}$$

where i is the iteration number, $\hat{L}_i = 2\hat{A}_i/\hat{\sigma}_i^2$ is the estimated channel reliability and where $\mathrm{dec}_\mathrm{A}(\cdot)$ and $\mathrm{dec}_\sigma(\cdot)$ are decision functions, given by either $\tanh(\cdot)$ or $\mathrm{sign}(\cdot)$. We will consider four different estimators, denoted SS, SH, HS and HH, where the first S or H indicates whether soft or hard information, respectively, is used for gain estimation, and the second S or H indicates whether soft or hard information, respectively, is used for estimating noise variance. Note that the SH estimator corresponds to the SEM algorithm applied to a scalar channel. The EM algorithm, on the other hand, cannot be expressed in this framework. Its channel gain estimator can be expressed as in (31), with $\mathrm{dec}_\mathrm{A}(\cdot) = \tanh_\mathrm{A}(\cdot)$. Its noise variance estimator, however, is given by

$$\begin{aligned}\hat{\sigma}_{i+1}^2 &= \frac{1}{N}\sum_{k=0}^{N-1}\left[|r_k|^2 + \hat{A}_{i+1}^2 - 2r_k\hat{A}_{i+1}\tanh\left(\frac{1}{2}\hat{L}_i r_k\right)\right] \\ &= \frac{1}{N}\sum_{k=0}^{N-1}|r_k|^2 - \hat{A}_{i+1}.\end{aligned} \tag{33}$$

Now suppose the number of observations tends to infinity. In this case, we may use the law of large numbers in (31), and (32). Thus, in this asymptotic case, the channel HH, SH, SS and HS estimators may be written as

$$\hat{A}_{i+1} = \mathrm{E}\left[r_k\mathrm{dec}_\mathrm{A}\left(\frac{1}{2}\hat{L}_i r_k\right)\right]. \tag{34}$$

$$\hat{\sigma}_{i+1}^2 = \mathrm{E}\left[\left|r_k - \hat{A}_{i+1}\mathrm{dec}_\sigma\left(\frac{1}{2}\hat{L}_i r_k\right)\right|^2\right]. \tag{35}$$

Equation (34) also describes the gain estimator of the EM algorithm. The noise variance estimator for the EM algorithm is obtained by applying the law of large numbers in (33), yielding

$$\hat{\sigma}^2_{i+1} = \mathrm{E}\left[|r_k|^2\right] - \hat{A}^2_{i+1} \tag{36}$$

$$= A^2 + \sigma^2 - \hat{A}^2_{i+1}. \tag{37}$$

For the HH estimator, it is shown in Appendix A that (34) and (35) may be written in closed form as

$$\hat{A}_{i+1} = A\left(1 - 2\,\mathrm{Q}\left(\frac{A}{\sigma}\right)\right) + \sqrt{\frac{2}{\pi}}\sigma\exp\left(-\frac{A^2}{2\sigma^2}\right), \tag{38}$$

and

$$\hat{\sigma}^2{}_{i+1} = A^2 + \sigma^2 - \hat{A}^2_{i+1}, \tag{39}$$

with $\sqrt{A^2+\sigma^2} \geq \hat{A} \geq A$. Therefore, for the HH estimator neither $\hat{A}_{i+1}$ nor $\hat{\sigma}^2{}_{i+1}$ depend on the iteration number i. Unfortunately, if soft information is used, equation (34) cannot be computed in closed form, so we must resort to numerical integration.

From (34), (35) and (37), we see that, as N tends to infinity, $\hat{A}_{i+1}$, and consequently $\hat{\sigma}^2_{i+1}$, is a function of just $\hat{L}_i$. The fact that both $\hat{A}_{i+1}$ and $\hat{\sigma}^2_{i+1}$ depend on a single parameter allows for a graphical study of the iterative process. This analysis is clearer if we consider the ratio $\alpha_i = \hat{L}_i / L$ instead of $\hat{L}_i$, where α_i is the relative estimated channel reliability, defined as the ratio between the estimated channel reliability at the i-th iteration and the actual channel reliability $L = 2A/\sigma^2$. For the graphical analysis, we view one iteration of the SEM algorithm as a function whose input is α_i and whose output is α_{i+1}.

This function is plotted in a graph, along with the line $\alpha_{i+1} = \alpha_i$. Since the algorithm converges when $\alpha_i = \alpha_{i+1}$, the fixed points of the SEM algorithm are given by the intersection of the two curves.

In Fig. 6 we plot α_{i+1} *versus* α_i for the five estimators, assuming $A = \sqrt{2}$ and an $SNR = A^2/\sigma^2 = 2$ dB. We also plot the line $\alpha_{i+1} = \alpha_i$, which allows for the graphical determination of the behavior of the algorithms as follows. Initially, at the zero-th iteration, a value α_0 is given. The estimator then produces a value of α_1, which can be determined graphically as shown by the vertical arrow in Fig. 6 for the EM algorithm and $\alpha_0 = 2.2$. The value of α_i for the next iteration can now be found by the vertical arrow in Fig. 6, which connects the point (α_0, α_1) to the point (α_1, α_1). Now the value of α_2 can be determined by a vertical line, not shown in Fig. 6, that connects the point (α_1, α_1) to the

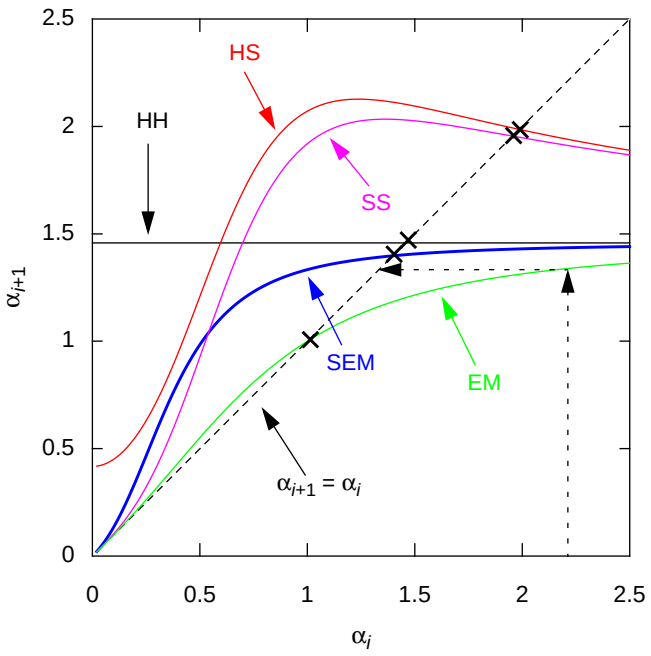

Fig. 6. Estimated relative channel reliability α_i as a function of its value in the previous iteration, α_{i-1}.

EM curve. The process then repeats. It is clear that the iterations stop when the curve for a given algorithm intersects the line $\alpha_{i+1} = \alpha_i$. The values for which this happens, α^*, are the fixed point of the algorithms and are marked with '×' in Fig. 6.

Some interesting observations can be made from Fig. 6. Consider, for instance, the HH estimator. For this estimator, we see in Fig. 6 that the value of α_{i+1} does not depend on α_i. Thus, following the iterative procedure, we see that the HH algorithm converges in a single iteration, as was expected from the analysis in (38) and (39). We can also see that the EM and the SEM algorithms generate a monotone sequence α_i. In other words, if these algorithms are initialized with an α_0 larger (smaller) than their fixed point α^*, then α_i will monotonically decrease (increase) until they converge. On the other hand, the α_i for the HS and SS algorithms eventually become greater than α^*. After that happens, they alternate between values that are greater than and smaller than α^*.

Using Fig. 6, we can determine the value of α after convergence for each algorithm. Then, we can use (34) and (35) to determine the expected values of $\hat{A}$ and $\hat{\sigma}$ after convergence. The resulting estimation errors are listed in Table 1. As we can see, the values in Table 1 indicate that the best strategy is the EM algorithm, and the SEM estimator produces the second best results.

Table 1:Expected Values of Estimation Error After Convergence

Estimator type	$\lvert\hat{A} - A\rvert^2$ (dB)	$\lvert\hat{\sigma} - \sigma\rvert^2$ (dB)
SS	-21.7	-10.5
SEM	-25.3	-16.2
HH	-19.1	-16.1
HS	-19.1	-10.4
EM	$-\infty$	$-\infty$

Even though the EM algorithm is expected to produce exact estimates, its convergence can be very slow. This can be seen in Fig. 7, where we plot the expected trajectories of the EM and the SEM algorithms, assuming that both algorithms are initialized using the HH estimates. The HH estimates are a good candidate for initialization: they have reasonable performance and converge in one iteration. As we can see in Fig. 7, the SEM estimator is expected to converge in roughly 2 iterations, while the EM estimator is expected to converge in roughly 7 iterations.

The performance of the estimators can be computed using the method described above for other values of SNR, yielding the plots of the estimation errors for $\hat{A}$ and $\hat{\sigma}^2$ *versus* SNR shown in the dashed lines in Fig. 8 and Fig. 9, respectively. Again, we see that the EM algorithm gives the best overall performance, followed by the SEM estimator. For comparison, we also show simulation results in Fig. 8 and Fig. 9. These correspond to the

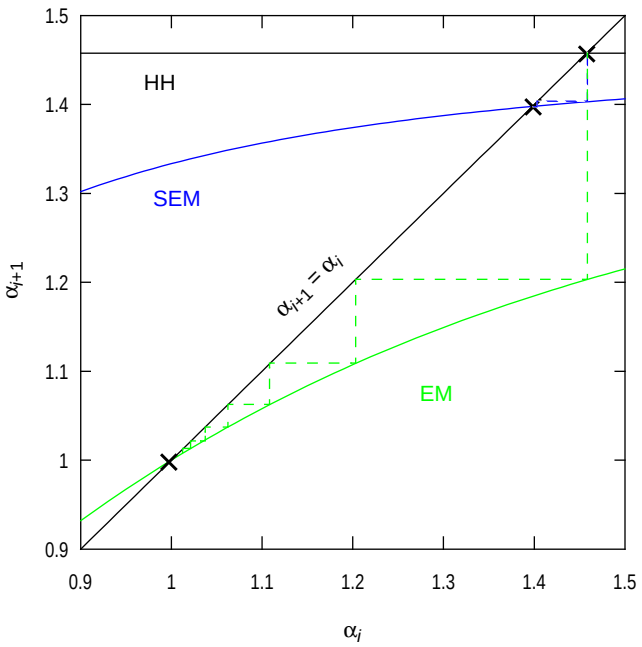

Fig. 7. Tracking the trajectories of the EM and the SEM estimators for a scalar channel.

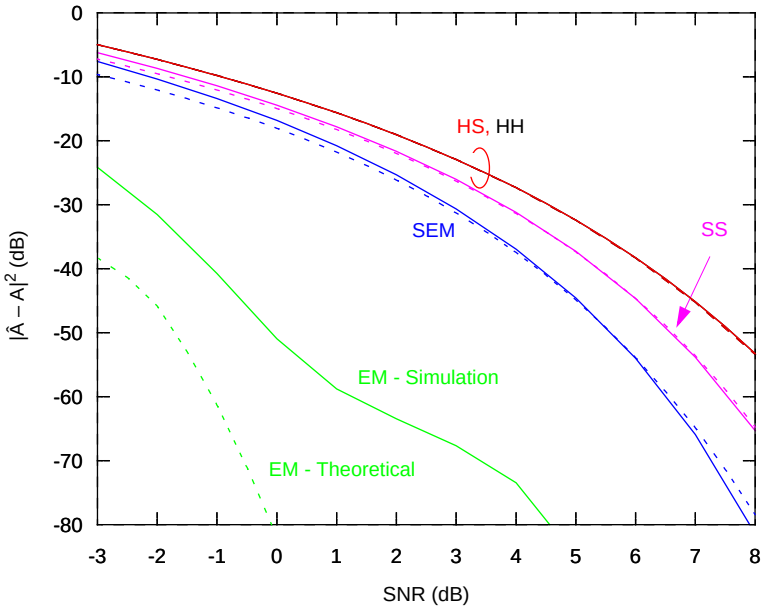

Fig. 8. Asymptotic error of gain estimates as a function of SNR. Dashed lines correspond to theoretical predictions, solid lines correspond to a simulation with 10^6 transmitted bits.

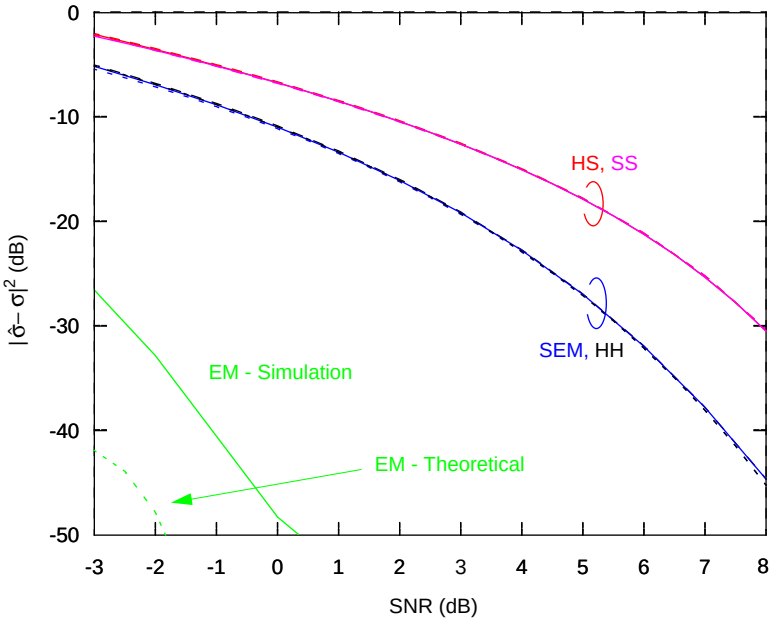

Fig. 9. Asymptotic error of noise variance estimates as a function of SNR. Dashed lines correspond to theoretical predictions, solid lines correspond to a simulation with 10^6 transmitted bits.

solid lines, and were obtained using 10^6 transmitted bits, and the channel estimators were run until $| \hat{L}_i - \hat{L}_{i-1} | < 10^{-5}$ or the number of iterations exceeded 20. As we can see, the theoretical curves predict the performance of the estimators very closely, except for the EM algorithm. An explanation for the difference between the theoretical and simulation curves for the EM algorithm could not be found.

3.3 The Impact of the Estimated Noise Variance

It is interesting to note that while substituting the actual values of $\boldsymbol{h}$ or $\boldsymbol{a}$ for their estimates will always improve the performance of the iterative algorithm, the same is not true for σ. Indeed, substituting σ for $\hat{\sigma}$ will often result in performance degradation. Intuitively, one can think of $\hat{\sigma}$ as playing two roles: in addition to measuring σ, it also acts as a *measure of reliability* in the channel estimate $\hat{\boldsymbol{h}}$. Consider a decomposition of the channel output:

$$r_k = \hat{\boldsymbol{h}}^{\mathrm{T}} \boldsymbol{a}_k + (\boldsymbol{h} - \hat{\boldsymbol{h}})^{\mathrm{T}} \boldsymbol{a}_k + n_k. \tag{40}$$

The term $(\boldsymbol{h} - \hat{\boldsymbol{h}})^{\mathrm{T}} \boldsymbol{a}_k$ represents the contribution to r_k from the estimation error. By using $\hat{\boldsymbol{h}}$ to model the channel in the BCJR algorithm, we are in effect lumping the estimation error with the noise. Combining the two results in an effective noise sequence with variance larger than σ^2. It is thus appropriate that $\hat{\sigma}$ should exceed σ whenever $\hat{\boldsymbol{h}}$ differs from $\boldsymbol{h}$. Alternatively, it stands to reason that an unreliable channel estimate should translate to an unreliable (*i.e.*, with small magnitude) symbol estimate, regardless of how well $\hat{\boldsymbol{h}}^{\mathrm{T}} \boldsymbol{a}_k$ matches r_k. Using a large value of $\hat{\sigma}$ in the BCJR equalizer ensures that its output will have a small magnitude. Fortunately, the noise variance estimate produced by (29) measures the energy of both the second and the third term in (40). If $\hat{\boldsymbol{h}}$ is a poor

channel estimate, $\tilde{\boldsymbol{a}}$ will also be a poor estimate for $\boldsymbol{a}$, and convolving $\tilde{\boldsymbol{a}}$ and $\hat{\boldsymbol{h}}$ will produce a poor match for $\boldsymbol{r}$, so that (29) will produce a large estimated noise variance.

3.4 Simulation Results

In section 3.2, we saw that the EM algorithm outperforms the SEM algorithm for a scalar channel and as the number of observations N tends to infinity. In this section, we present simulation results showing that the performance degradation incurred by ignoring the matrix $\hat{\mathbf{R}}_{\boldsymbol{a}}$ in the EM algorithm is not significant for finite N and a channel that introduces ISI. We used the simulation scenario of [22]. The channel is given by $\boldsymbol{h}$ = [0.5 0.7 0.5], and the noise variance is chosen so that SNR = 11 dB, where $SNR = \|\boldsymbol{h}\|^2/\sigma^2$. We initialized the estimates to $\hat{\boldsymbol{h}}_0 = [0, \hat{\sigma}_0, 0]$, and

$$\hat{\sigma}_0^2 = \frac{1}{2N}\sum_{k=0}^{N-1}|r_k|^2. \tag{41}$$

Thus, we have initialized our estimate of the SNR to 0 dB, and the values of $\hat{\boldsymbol{h}}_0$ and $\hat{\sigma}_0$ agree with the energy of the received signal. In Fig. 10, we show the estimates of the

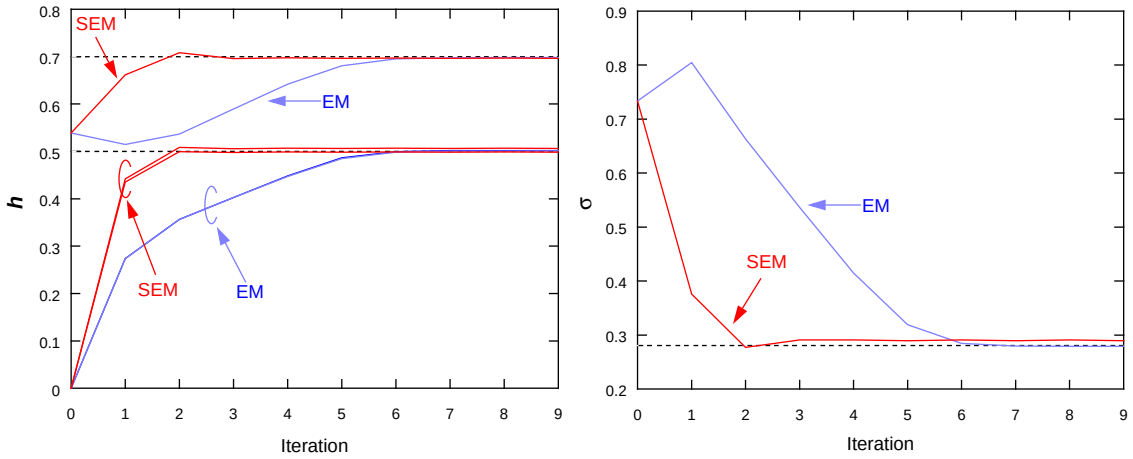

Fig. 10. Performance comparison: channel and noise standard deviation estimates as a function of iteration for EM (light solid) and simplified EM (solid)

channel coefficients and the noise standard deviation as a function of iteration for the EM and the SEM algorithms, averaged over 100 independent blocks of 256 BPSK symbols. As expected, the SEM algorithm yields a larger estimation error than the EM algorithm, though the performance loss is not significant. As in the scalar channel case, the SEM algorithm converges faster than EM in this experiment.

In this chapter, we only simulated the performance of the SEM algorithm for one ISI channel. In Chapter 4, we introduce a modification to the SEM algorithm that greatly improves its convergence. More simulations will be conducted then.

3.5 Summary

In this section, we proposed the SEM algorithm, an iterative blind channel estimator that is less complex than the EM algorithm in two ways: it does not require the computation and inversion of the autocorrelation matrix, and it is not intrinsically tied to the BCJR equalizer. We presented an asymptotic analysis of different estimators, including the SEM and EM algorithms, for a scalar channel and as the number of observations tends to infinity. We showed that the EM algorithm provides the best estimates in this case, followed by the SEM algorithm. We also showed that for a scalar channel the SEM algorithm is expected to converge faster than the EM algorithm. For a channel that introduces ISI, simulation results indicate that the performance loss of the SEM is not significant when compared to the EM algorithm, and that the SEM estimates converge faster than the EM estimates.

CHAPTER 4

The Extended-Window Algorithm (EW)

As we discussed in section 2.3, the EM algorithm generates a sequence of estimates with nondecreasing likelihood. Thus, it is prone to misconvergence, defined in the present context as the convergence to a nonglobal local maximum of the likelihood function. The traditional approach to this problem is either to completely ignore misconvergence or to assume the availability of a good initialization. For instance, the simulation in the previous section involved some cheating: the channel estimates were initialized to an impulse at the center tap, which happens to match the main tap of the channel. However, there is no reason for using such initialization other than the fact that we know that the center tap of the actual channel is dominant, a knowledge that obviously would not be available in a real-world blind application. In this chapter, we show that the estimates after misconvergence may have a structure that allows some local maxima to be escaped.

4.1 A Study of Misconvergence

To study an example of misconvergence, consider using the SEM algorithm to identify the maximum-phase channel $\boldsymbol{h} = [1\ 2\ 3\ 4\ 5]^T$ at $SNR = 24$ dB, with a BPSK input sequence. With the channel estimates being initialized to $\hat{\boldsymbol{h}}_0 = [1\ 0\ 0\ 0\ 0]^T$ and $\hat{\sigma}_0 = 1$, after 20 iterations the SEM algorithm converged to a fixed point of

$$\hat{\boldsymbol{h}} = [2.1785 \quad 3.0727 \quad 4.1076 \quad 5.0919 \quad 0.1197]^{\mathrm{T}}. \tag{42}$$

The algorithm thus fails. But the estimated channel is roughly a shifted version of $\boldsymbol{h}$. A possible explanation for this behavior is that, if the channel is not minimum phase, then it introduces some delay δ that cannot be compensated for at the symbol estimator of Fig. 5. Thus, the soft symbol estimate L_k produced by the symbol estimator may in fact be related to a delayed symbol $a_{k-\delta}$, *i.e.*, $L_k \approx \log \Pr(a_{k-\delta} = +1 \mid \boldsymbol{r}) / \Pr(a_{k-\delta} = -1 \mid \boldsymbol{r})$. Therefore, when using equation (28) to estimate h_n, we may be estimating $h_{n+\delta}$ instead. In this example, the delay is 1. Apart from this delay, the algorithm seems to perform well, and in fact if it were to also compute

$$\mathrm{E}\left[r_k \tanh\left(\frac{L_{k+1}}{2}\right)\right], \tag{43}$$

it would also be able to accurately estimate h_0. Hence, to estimate all the channel coefficients in this example, we must compute equation (28) for more values than originally suggested by the EM algorithm.

For a general channel, we have observed that, after convergence, the sign of the LLR produced by the BCJR algorithm is related to the actual symbol a_k by $\mathrm{sign}(L_k) \approx a_{k-\delta}$, for some integer delay δ satisfying $|\delta| \leq \mu$. Even though a proof of this bound for the delay could not be obtained, there is an intuitive explanation for this behavior. Let $d = \mathrm{argmax}_{0 \leq j \leq \mu} |h_j|$. If the actual channel were known to the symbol estimator, then r_k is the channel output that has the largest impact on the decision made on a_{k-d}. Now assume that the channel estimator passes $\hat{\boldsymbol{h}}$ to the symbol estimator, and let $\hat{d} = \mathrm{argmax}_{0 \leq j \leq \mu} |\hat{h}_j|$. With this channel estimate, the symbol estimator will be such that r_k is the channel output that has the largest impact on the decision made on $a_{k-\hat{d}}$. If this estimate is

reasonable, then $\hat{a}_{k-\hat{d}} \approx a_{k-d}$, since they are both mostly influenced by r_k. In other words, after convergence, $\text{sign}(L_k) \approx a_{k-(d-\hat{d})}$. Now let $\delta = d - \hat{d}$. Since $d, \hat{d} \in \{0, \ldots \mu\}$, we indeed have $|\delta| \leq \mu$.

To illustrate the effects of the delay in channel estimation, consider for instance the channel $\boldsymbol{h} = [1\ \varepsilon\ \varepsilon\ \varepsilon\ \varepsilon]^T$ for some small ε, and assume that the channel estimator passes $\hat{\boldsymbol{h}} = [0\ 0\ 0\ 0\ 1]^T$ and a given $\hat{\sigma}^2$ to the symbol estimator. With these values, the output of the symbol estimator would essentially be $L_k = 2r_{k+4}/\hat{\sigma}^2$. But we know that, for the channel $\boldsymbol{h}$, $\text{sign}(r_k) \approx a_k$, and hence $\text{sign}(L_k) \approx a_{k+4}$. Thus, if we compute (28) for $n = -4, \ldots 0$, as originally suggested, we will never get a chance to compute, for instance,

$$\hat{h}_1 = \frac{1}{N}\sum_{k=1}^{N} r_k a_{k-1} \approx \frac{1}{N}\sum_{k=1}^{N} r_k \tanh(L_{k-5}/2). \tag{44}$$

Likewise, if $\boldsymbol{h} = [\varepsilon\ \varepsilon\ \varepsilon\ \varepsilon\ 1]^T$, and the channel estimator passes $\hat{\boldsymbol{h}} = [1\ 0\ 0\ 0\ 0]^T$ and a given $\hat{\sigma}^2$ to the symbol estimator, then we would have $L_k = 2r_k/\hat{\sigma}^2$, so that $\text{sign}(L_k) \approx a_{k-4}$. Thus, if we compute (28) for the given window $n = -4, \ldots 0$, we would never get a chance to estimate $\hat{h}_1$. For that, we would have to use L_{k+3}. Even though these are extreme examples, they illustrate well the effects of the delay in the iterative process.

4.2 The EW Channel Estimator

In light of the discussion above, it is clear that if we are to correctly estimate the channel, we cannot restrict the computation of (28) to the window $n = -\mu, \ldots 0$. Thus, we propose an *extended window* EM (EW) algorithm. To determine how much the window must be extended, we again consider the extreme cases. When $\text{sign}(L_k) \approx a_{k-\mu}$, to

estimate h_0 and h_μ we need to compute (28) for $n = -\mu$ and $n = 0$, respectively. Likewise, when $\text{sign}(L_k) \approx a_{k+\mu}$, to estimate h_0 and h_μ we need to compute (28) for $n = \mu$ and $n = 2\mu$, respectively. Thus, we propose to compute an auxiliary vector $\boldsymbol{g}$ as

$$g_n = \frac{1}{N}\sum_{k=1}^{N}\left[r_k \tanh\left(\frac{L_{k-n}}{2}\right)\right], \text{ for } n = -\mu, \ldots, 2\mu. \tag{45}$$

Note that only $\mu+1$ adjacent elements of $\boldsymbol{g}$ are expected to be non-zero. With that in mind, we propose that the channel estimates $\hat{\boldsymbol{h}}$ be the $\mu+1$ adjacent elements of $\boldsymbol{g}$ with highest energy.

4.2.1 Delay and Noise Variance Estimator

Let $\hat{\boldsymbol{h}} = [g_{-\delta}, \ldots g_{-\delta+\mu}]^T$ be the portion of $\boldsymbol{g}$ with largest energy. Note that after convergence we expect that $\hat{\boldsymbol{h}} = \boldsymbol{h}$, *i.e.*, $g_{-\delta} = h_0$. But comparing (25) and (45), we note that this is equivalent to saying that

$$a_k \approx \tanh\left(\frac{L_{k+\delta}}{2}\right). \tag{46}$$

In other words, by choosing $\hat{\boldsymbol{h}}$ to be the current channel estimate we are inherently assuming that the estimated sequence is a delayed version of the transmitted one, where the delay is δ. This delay should be taken into account in the estimation of the noise variance. With that in mind, we propose to estimate σ^2 using a modified version of (29), namely

$$\hat{\sigma}^2_{i+1} = \frac{1}{N}\sum_{k=1}^{N}\left|r_k - \hat{\boldsymbol{a}}^T_{k-\delta}\hat{\boldsymbol{h}}_{i+1}\right|^2. \tag{47}$$

The EW algorithm is summarized in the following pseudocode:

```
initialize channel estimates ĥ_0 and σ̂_0^2.
i = 0.
repeat
      use current channel estimates to compute symbol estimates
            L_k, for k = 0, … N-1.
      Compute g_n, n = –μ, … 2μ, as in (45).
      Let ĥ_i = [g_–d, … g_–d + μ]^T be the μ+1 consecutive entries
            of g with highest energy.
      Update the noise variance estimate according to (47).
      i = i + 1.
until a stop criterion is met
```

4.3 Simulation Results

In this section, we present some simulation results comparing the performance of the EW algorithm to the EM algorithm and to trained channel estimation algorithms. In all the simulations, we have used a BCJR equalizer with the EW algorithm, to allow for a fair comparison with the EM algorithm. The results presented in this section all correct for the aforementioned delays in the channel estimation process. In other words, when computing estimation error or averaging channel estimates, the estimates were shifted to best match the actual channel. Note that this does not affect the channel estimates in the iterative procedure.

As a first test of the extended-window algorithm, we simulated the transmission of K = 600 bits over the channel $\boldsymbol{h}$ = [-0.2287, 0.3964, 0.7623, 0.3964, -0.2287] $^{\mathrm{T}}$ from [26], whose frequency response is shown in Fig. 11. We have used SNR = $\|\boldsymbol{h}\|^2/\sigma^2$ = 9 dB. To stress the fact that the proposed algorithm is not sensitive to initial conditions, we initialized $\hat{\boldsymbol{h}}$ randomly using $\hat{\boldsymbol{h}}_{(0)} = \boldsymbol{u}\hat{\sigma}_{(0)}/\|\boldsymbol{u}\|$, where $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\hat{\sigma}^2_{(0)} = \sum_{k=0}^{N-1} r_k^2/2N$. (This implies an initial estimated SNR of 0 dB, with values consistent with the received energy.) In Fig. 12, we show the convergence behavior of the SEM

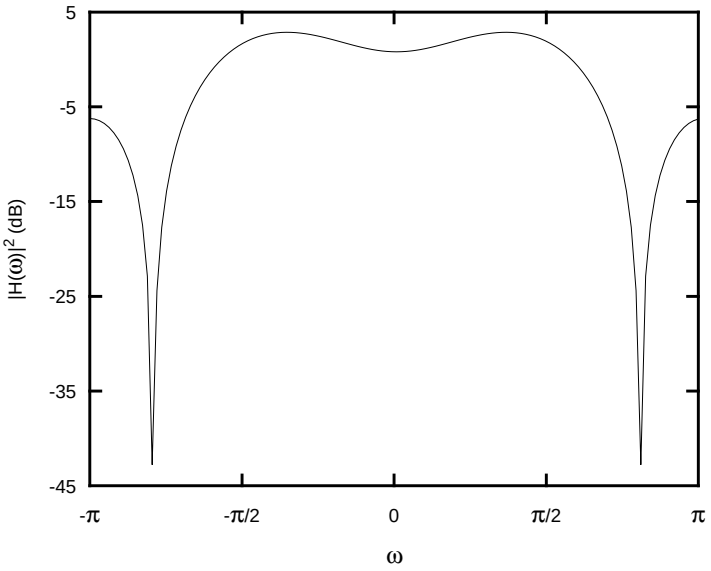

Fig. 11. Frequency response of h = [–0.2287, 0.3964, 0.7623, 0.3964, –0.2287].

channel estimates, averaged over 200 independent runs of this experiment. Only the convergence of $\hat{h}_0$, $\hat{h}_1$ and $\hat{h}_2$ is shown; the behavior of $\hat{h}_3$ and $\hat{h}_4$ is similar to that of $\hat{h}_2$ and $\hat{h}_0$, respectively, but we show only those with worse convergence. The shaded regions around the channel estimates correspond to plus and minus one standard deviation. For comparison, we show the average behavior of the EM estimates in Fig. 13.

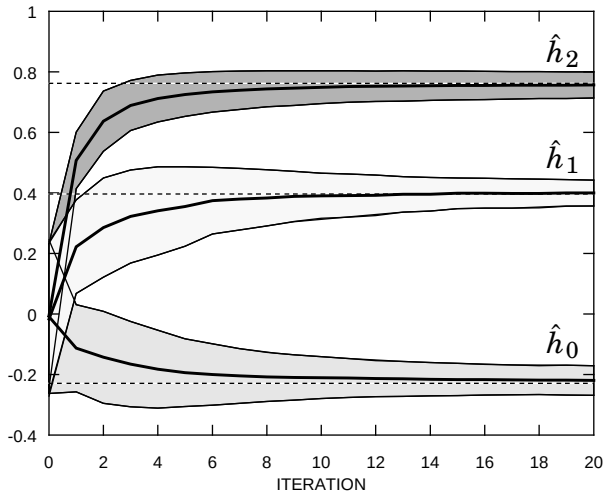

Fig. 12. Estimates of $\boldsymbol{h}$ = [–0.2287, 0.3964, 0.7623, 0.3964, –0.2287], produced by the extended-window algorithm.

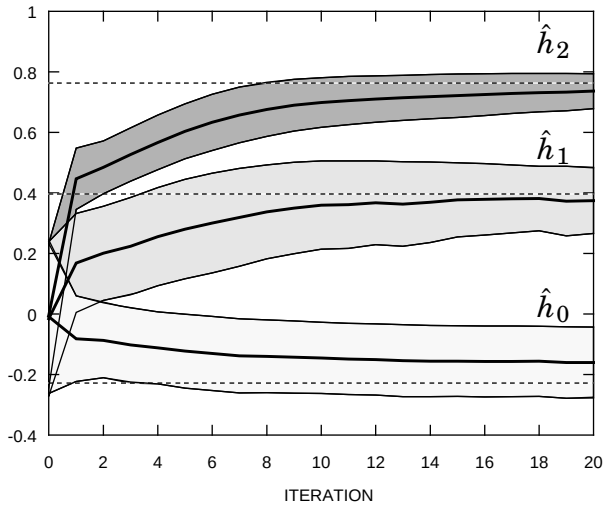

Fig. 13. EM estimates of $\boldsymbol{h}$ = [–0.2287, 0.3964, 0.7623, 0.3964, –0.2287].

Unlike the good performance of the extended window algorithm, the EM algorithm even fails to converge in the mean to the correct estimates, especially $\hat{h}_0$. This happens because the EM algorithm gets trapped in local maxima of the likelihood function [40], while the extended-window avoids many of these local maxima. The better convergence behavior of the EW algorithm is even more clear in Fig. 14, where we show the noise variance estimates.

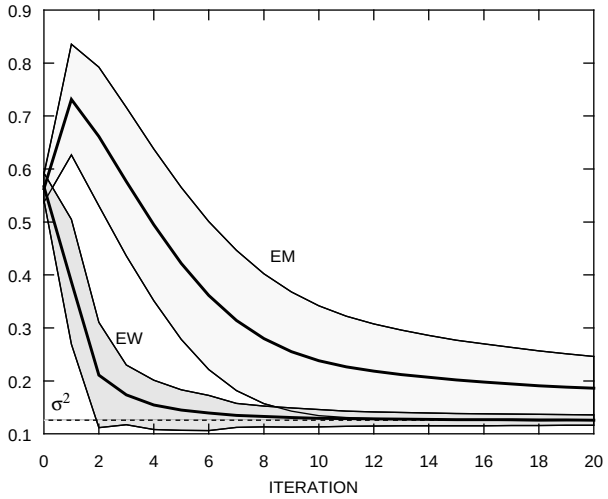

Fig. 14. Estimates of σ^2, produced by the extended-window algorithm.

The performance of the EW estimator was also compared to a trained estimator that estimates the channel coefficients using equation (28) with the actual transmitted symbol a_{k-n} substituting the estimate $\tilde{a}_{k-n}^{(i)}$. This trained estimation technique, known as channel probing, is not the trained MMSE estimator of (17) and (18). For comparison purposes, we also show the performance of the EM algorithm and the trained MMSE estimator. We have simulated the transmission of 200 blocks of $K = 600$ bits over the channel $\boldsymbol{h} = [-0.2287, 0.3964, 0.7623, 0.3964, -0.2287]^{\mathrm{T}}$. For each block, the channel estimates for the EM and EW algorithms were initialized with the random estimates used in the previous experiment.

In Fig. 15 we show the estimation error as a function of the SNR for the trained estimates and for the EM and EW estimates after 20 iterations. In Fig. 16, we show the resulting BER. Again we see that the EW algorithm performs better than the EM algorithm. It is interesting to notice that the performance of the EW algorithm approaches that of its trained counterpart, the channel probing estimator. One would thus expect the

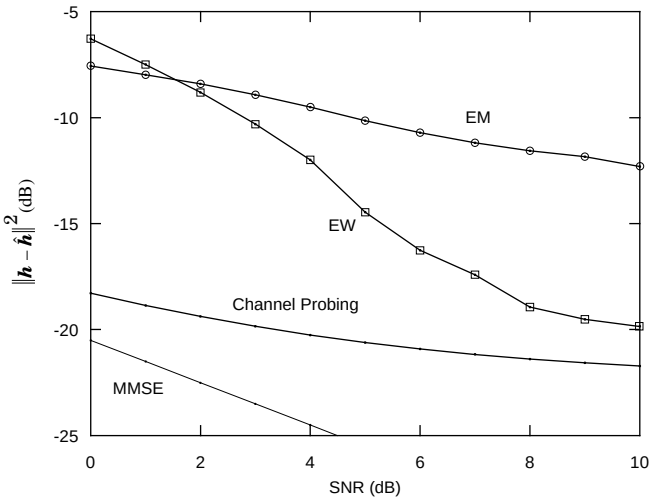

Fig. 15. Estimation error for the channel probing, MMSE, EM and EW estimates after 20 iterations.

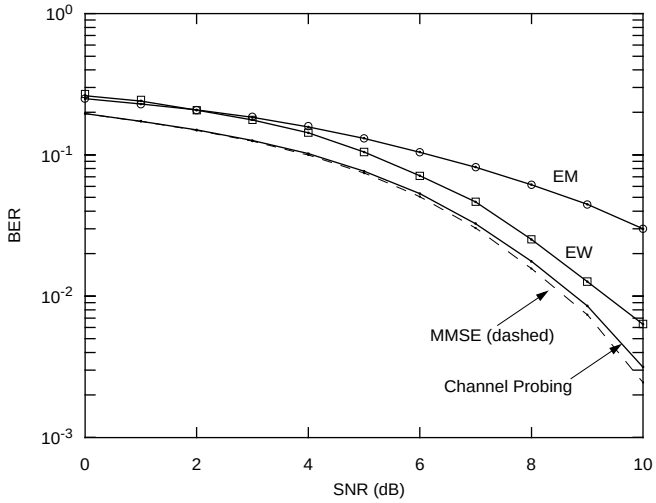

Fig. 16. Bit error rate using the trained, EM and EW estimates after 20 iterations.

performance of the EM algorithm to approach that of its trained counterpart, the MMSE algorithm. However, as we can see from Fig. 15 and Fig. 16, the EM algorithm performs worse than channel probing, which is in turn worse than the MMSE estimator. Finally, it should be pointed out that even though the channel estimates provided by the MMSE algorithm are better than those of the channel probing, the BER of both estimates is similar. In other words, the channel probing estimates are "good enough", and the added complexity of the MMSE estimator does not have much impact on the BER performance in the SNR range considered here.

To further support the claim that the proposed algorithm avoids most of the local maxima of the likelihood function that trap the EM algorithm, we ran both algorithms on 1,000 random channels of memory $\mu = 4$, generated as $\boldsymbol{h} = \boldsymbol{u}/\|\boldsymbol{u}\|$, where $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The estimates were initialized to $\hat{\sigma}_0^2 = \sum_{k=0}^{N-1} r_k^2 / 2N$ and $\hat{\boldsymbol{h}}_0 = (0, 0, \hat{\sigma}_0, 0, 0)^T$, *i.e.*, the center tap of $\hat{\boldsymbol{h}}_0$ is initialized to $\hat{\sigma}_0$. We used SNR = 18 dB, and blocks of $K = 1000$ bits.

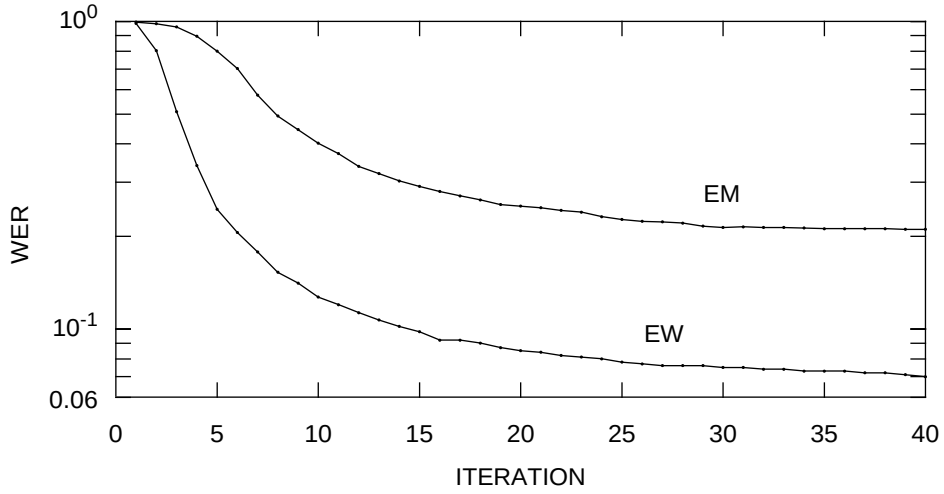

Fig. 17. WER for the EW and the EM algorithms for an ensemble of 1,000 random channels.

In Fig. 17 we show the word error rate (WER) (percentage of blocks detected with errors) of the EW and EM algorithms versus iteration. It is again clear that the EW algorithm has a better performance than the EM algorithm. This can also be seen in Fig. 18, where we show histograms of the estimation errors (in dB) for the channel probing, EW, and EM estimates, computed after 40 iterations. We see that while only 3%

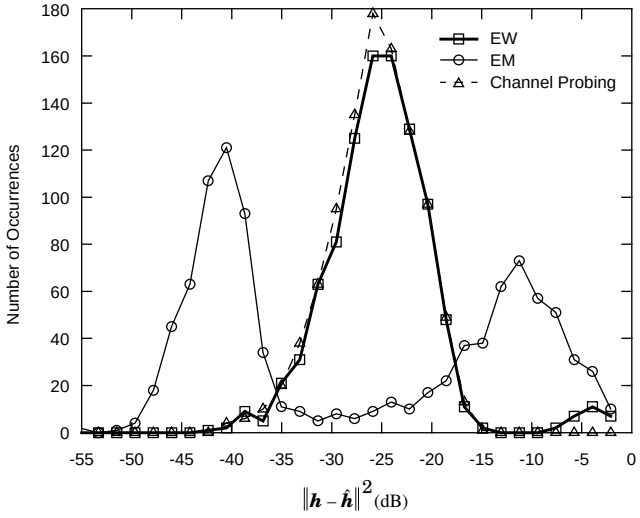

Fig. 18. Histograms of estimation errors for the EW and the EM algorithms over an ensemble of 1,000 random channels.

of the EW estimates have an error larger than -16 dB, 35% of the EM estimates have an error larger than -16 dB. In fact, the histogram for the EW algorithm is very similar to that of the channel probing estimates, which again shows the good convergence properties of the EW algorithm.

It is also interesting to note in Fig. 18 that the EM estimates have a bimodal behavior: the estimation errors produced by the EM algorithm are grouped around -11 dB and -43 dB. These groups are respectively better than and worse than the channel probing estimates. This bimodal behavior can be explained by the fact that the EM algorithm converges to inaccurate estimates very often, leading to large estimation errors. On the other hand, when the EM algorithm converges to accurate estimates, then the EM estimates are close to the MMSE estimates, which are better than those produced by channel probing. However, as we previously observed, the better quality of the channel estimates has no significant impact on the BER performance: the equalizer based on the channel probing estimates detected all transmitted sequences correctly.

4.4 Summary

In this chapter, we studied some aspects of the convergence of the EM and the SEM algorithm. We showed that the EM and SEM estimates after misconvergence may be a shifted version of the channel. With that in mind, we proposed the EW algorithm, a modification of the SEM algorithm that exploits the structure of the estimates after misconvergence to greatly decrease the probability of misconvergence. We showed via simulations that the EW algorithm has better convergence properties than the EM algorithm when the initialization and/or the channel is random, yielding a better

performance both in terms of BER and channel estimation error. In simulations, we also compared the performance of the EW algorithm to a system that estimates the channel using channel probing and knowledge of the transmitted symbols. We showed that the performance gap between the EW system and the one with training is surprisingly small. It should be pointed out that the trained estimators used in this section are unrealistic, since they assume knowledge of the whole transmitted sequence. Therefore, the gap between the EW and trained estimates should be even smaller in a practical system.

CHAPTER 5

The Soft-Feedback Equalizer

In section 2.3, the EM algorithm was shown to have three problems: slow convergence, misconvergence and high computational complexity, stemming from need to compute and invert the *a posteriori* sample autocorrelation matrix $\hat{\mathbf{R}}_{\boldsymbol{a}}$ defined in (19) and from the use of the BCJR equalizer. In chapter 3, we proposed the SEM algorithm, which converges faster than the EM algorithm and does not require matrix inversion. In chapter 4, we proposed the EW algorithm, which decreases the probability of misconvergence. In this chapter, we will address the remaining problem of the EM algorithm: the complexity that results from using the BCJR algorithm for estimating the transmitted symbol.

The SEM and EW algorithms are not intrinsically tied to the BCJR equalizer, and may be used with any equalizer that produces soft symbol information. With that in mind, in this chapter we propose the soft decision feedback equalizer with priors (SFE), a low complexity alternative to the BCJR algorithm that retains many of its attractive features. In particular, it outputs soft information in the form of an estimate of the *a posteriori* LLR. It also exploits *a priori* information on the transmitted symbols, in the form *a priori* LLR λ_k^p. Thus, it is well suited for applications such as turbo equalization (where the *a priori* probabilities are provided by the channel decoder), semi-blind systems (in which the *a*

priori probabilities stem from the fact that some symbols are known), and iterative equalization (in which the *a priori* probabilities come from a previous iteration of the equalizer).

The SFE will be derived in a general context, in which we assume the availability of channel estimates and *a priori* probabilities. That way, the SFE is not tied to a single application, such as iterative channel estimation or turbo equalization.

5.1 Previous Work on Interference Cancellation

The reduced-complexity equalization techniques proposed in [30-35], which are based on linear filters and exploit *a priori* information, have the structure shown in Fig. 19. In this figure, the received signal is filtered by a linear filter $\boldsymbol{f}$, whose output contains residual ISI. The *a priori* information is used to estimate and cancel this residual ISI.

Interference cancellation (IC) proceeds as follows. Assume that, at time k, the equalizer seeks to estimate a_k. The *a priori* information is used to produce soft estimates $\{\tilde{a}_{l\neq k}\}$ of the interfering symbols $\{a_{l\neq k}\}$, according to:

$$\tilde{a}_l = \mathrm{E}[a \mid \lambda_l^p] = \tanh(\lambda_l^p/2)\,. \tag{48}$$

If these estimates are correct, their effect on the output of $\boldsymbol{f}$ can be estimated and cancelled through linear filtering and subtraction. Specifically, as shown in Fig. 19, an interference

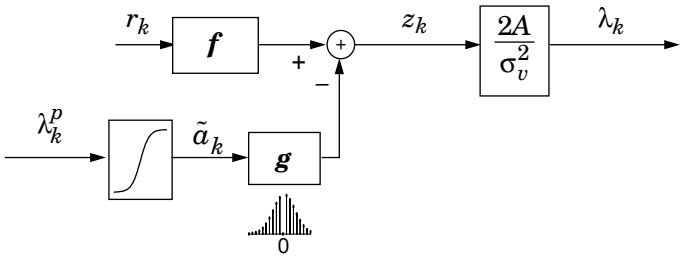

Fig. 19. Interference canceller with *a priori* information.

canceller feeds the soft decisions through a filter $\boldsymbol{g}$, whose response is related to the residual ISI at the output of $\boldsymbol{f}$. Since the equalizer output will be used to estimate a_k, the influence of a_k on the equalizer output should not be cancelled. Hence, the zero-th coefficient of $\boldsymbol{g}$ is constrained to be zero. The equalizers of [30-35] choose $\boldsymbol{g}$ under the assumption that its input symbol estimates are correct, yielding $g_k = \sum_l h_l f_{k-l}$ when $k \neq 0$.

Since the zero-th tap of $\boldsymbol{g}$ is zero, the equalizer output at time k, z_k, is not a function of λ_k^p. Thus, z_k can only be used to produce extrinsic information, which can be done by writing the equalizer output as

$$z_k = Aa_k + v_k, \tag{49}$$

where $A = \mathrm{E}[z_k a_k] = \sum_l h_l f_{-l}$, and v_k includes the effect of channel noise and residual ISI. Note that, from this definition, v_k is independent of a_k. The computation of the extrinsic LLR λ_k from z_k is easy when v_k is approximated by a Gaussian random variable with variance σ_v^2. In this case, we find that

$$\lambda_k = 2Az_k/\sigma_v^2. \tag{50}$$

The full LLR at the equalizer output is then given by $L_k = \lambda_k + \lambda_k^p$.

The equalizers proposed in [30] and [31], which we refer to as decision-aided equalizers (DAE), choose $\boldsymbol{f}$ under the assumption that the soft decisions of (48) are correct, which leads to the matched-filter solution $f_k = h_{-k}$. In the equalizers proposed in [32,34] and in one of those in [35], $\boldsymbol{f}$ is an MMSE-LE. This equalizer depends on λ_k^p and must be recomputed every time instant k, resulting in a time-varying equalizer (TVE) whose computational complexity is quadratic in the number of equalizer coefficients. Also proposed in [32,35] are approximations that yield time-invariant filters $\boldsymbol{f}$ and $\boldsymbol{g}$. In

particular, the switched-equalizer (SE) strategy proposed in [35] chooses $\boldsymbol{f}$ as either a traditional MMSE equalizer or a matched filter (MF). A simple criterion to choose between these two equalizers is proposed that depends on the quality of the *a priori* information. In all cases [30-35], the cancellation filter $\boldsymbol{g}$ is designed under the assumption that its input symbol estimates are correct, namely $g_k = \sum_l h_l f_{k-l}$ for $k \neq 0$.

5.2 The Soft-Feedback Equalizer

We now propose the SFE, a soft-output equalization scheme that shares many similarities with the interference-cancellation schemes of [30-35]; however, our approach differs in two substantial ways:

- At time k, when computing z_k, the previous equalizer outputs $\{\lambda_{k-j} : j > 0\}$ are known. With these values, we may compute the full LLR $L_{k-j} = \lambda^p_{k-j} + \lambda_{k-j}$, which provides a better estimate of a_{k-j} than λ^p_{k-j} alone. Thus, instead of using $\tilde{a}_{k-j}$ to cancel interference, we propose to use $\bar{a}_{k-j} = \mathrm{E}[a_{k-j} | L_{k-j}]$ for $j > 0$. This is similar in spirit to the principle behind a DFE. A DFE-based system is also proposed in [35]. However, the system of [35] feeds back hard decisions on the equalizer output, without combining them with the *a priori* information, and it performs worse than the systems without feedback described in section 5.1. In [20], a DFE was proposed to be used with *a priori* information. However, it does not use the *a priori* information to cancel post-cursor ISI, and it computes the DFE coefficients assuming correct decisions.

- As in [32-35], instead of trying to cancel all the interference, we pass $\bar{a}_k$ and $\tilde{a}_k$ through linear filters whose coefficients, along with $\boldsymbol{f}$, are computed to minimize

the MSE $\mathrm{E}[|z_k - a_k|^2]$. However, following [39,45], we use a Gaussian approximation to λ_k^p and λ_k that leads to time-invariant SFE coefficients, and hence to a per-symbol computational complexity that is proportional to the number of equalizer coefficients, as opposed to the quadratic per-symbol complexity of the TVE.

Applying the above two changes to Fig. 19 leads to the proposed SFE structure shown in Fig. 20, where the filters $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are strictly anticausal and strictly causal, respectively, and the filters $\boldsymbol{f}$, $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are chosen to minimize the MSE. The thicker line in the feedback loop represents the only actual change from Fig. 19.

5.2.1 The SFE Coefficients

Let the SFE output z_k be written as

$$z_k = \boldsymbol{f}^{\mathrm{T}} \boldsymbol{r}_k - \boldsymbol{g}_1^{\mathrm{T}} \tilde{\boldsymbol{a}}_k - \boldsymbol{g}_2^{\mathrm{T}} \bar{\boldsymbol{a}}_k , \tag{51}$$

where $\boldsymbol{f} = [f_{-M_1}, \ldots f_{M_2}]^{\mathrm{T}}$, $\boldsymbol{r}_k = [r_{k+M_1}, \ldots r_{k-M_2}]^{\mathrm{T}}$, $\boldsymbol{g}_1 = [g_{-M_1}, \ldots g_{-1}]^{\mathrm{T}}$, $\boldsymbol{g}_2 = [g_1, \ldots g_{M_2+\mu}]^{\mathrm{T}}$, $\tilde{\boldsymbol{a}}_k = [\tilde{a}_{k+M_1}, \ldots \tilde{a}_{k+1}]^{\mathrm{T}}$, $\bar{\boldsymbol{a}}_k = [\bar{a}_{k-1}, \ldots \bar{a}_{k-(M_2+\mu)}]^{\mathrm{T}}$, the superscript T denotes transpose, and M_1 and M_2 determine the lengths of the filters. Now, assume that $\mathrm{E}[\tilde{a}_k a_j] = \mathrm{E}[\bar{a}_k a_j] = \mathrm{E}[\tilde{a}_k \bar{a}_j] = 0$ when $k \neq j$. This seems reasonable, since $\tilde{a}_k$ and $\bar{a}_k$ are

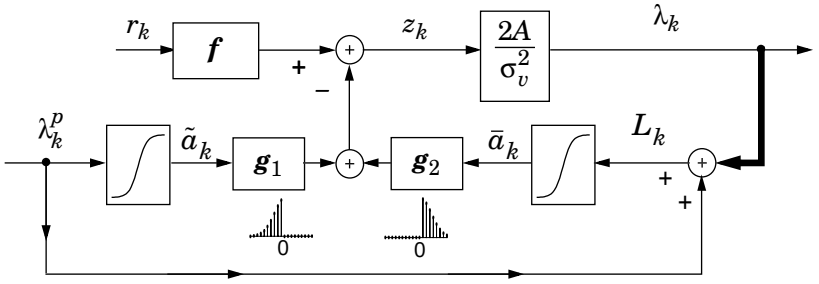

Fig. 20. The proposed SFE equalizer. The thicker line in the feedback loop represents the only actual change from Fig. 19.

approximately equal to a_k and the transmitted symbols are uncorrelated. Then, as shown in Appendix B, the values of $\boldsymbol{f}$, $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ that minimize $\mathrm{E}[\,|z_k - a_k|^2]$ are given by

$$\boldsymbol{f} = (\mathbf{H}\mathbf{H}^{\mathrm{T}} - \frac{\alpha_1^2}{E_1}\mathbf{H}_1\mathbf{H}_1^{\mathrm{T}} - \frac{\alpha_2^2}{E_2}\mathbf{H}_2\mathbf{H}_2^{\mathrm{T}} + \sigma^2\mathbf{I})^{-1}\,\boldsymbol{h}_0 \tag{52}$$

$$\boldsymbol{g}_1 = (\alpha_1/E_1)\,\mathbf{H}_1^{\mathrm{T}}\boldsymbol{f} \tag{53}$$

$$\boldsymbol{g}_2 = (\alpha_2/E_2)\,\mathbf{H}_2^{\mathrm{T}}\boldsymbol{f}, \tag{54}$$

where $\mathbf{H}$ is the $M \times (M + \mu)$ channel convolution matrix:

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & \dots & h_\mu & 0 & 0 & \dots & 0 \\ 0 & h_0 & h_1 & \dots & h_\mu & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & h_0 & h_1 & \dots & h_\mu \end{bmatrix}, \tag{55}$$

$M = M_1 + M_2 + 1$, and

$$E_1 = \mathrm{E}[\,|\tilde{a}_k|^2], \tag{56}$$

$$E_2 = \mathrm{E}[\,|\bar{a}_k|^2], \tag{57}$$

$$\alpha_1 = \mathrm{E}[\tilde{a}_k a_k], \tag{58}$$

$$\alpha_2 = \mathrm{E}[\bar{a}_k a_k]. \tag{59}$$

The vector $\boldsymbol{h}_0$ is the 0-th column of $\mathbf{H}$, where the columns of $\mathbf{H}$ are numbered as $\mathbf{H} = [\boldsymbol{h}_{-M_1}, \dots \boldsymbol{h}_{M_2+\mu}]$. Also, $\mathbf{H}_1 = [\boldsymbol{h}_{-M_1}, \dots \boldsymbol{h}_{-1}]$ and $\mathbf{H}_2 = [\boldsymbol{h}_1, \dots \boldsymbol{h}_{M_2+\mu}]$. The constants A and σ_v^2, needed in (50) to compute the LLR λ_k from the equalizer output, are shown in Appendix B to be given by $A = \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0$ and $\sigma_v^2 = A\,(1 - A)$. Thus,

$$\lambda_k = 2\,z_k/(1 - \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0). \tag{60}$$

Note that, from the definitions of $\tilde{\boldsymbol{a}}_k$ and $\bar{\boldsymbol{a}}_k$, at time k we only attempt to cancel the interference from the M_1 future symbols and the $M_2 + \mu$ past symbols. However,

$$\boldsymbol{r}_k = \mathbf{H}\boldsymbol{a}'_k + \boldsymbol{n}_k, \tag{61}$$

where $\boldsymbol{n}_k = [n_{k+M_1}, \ldots n_{k-M_2}]^{\mathrm{T}}$ and $\boldsymbol{a}'_k = [a_{k+M_1}, \ldots a_{k-M_2-\mu}]^{\mathrm{T}}$. Thus, the output of the linear filter $\boldsymbol{f}$ suffers the interference of a_{k-j} for $j = -M_1, \ldots M_2 + \mu$. In other words, at time k the output of $\boldsymbol{f}$ has residual interference from the M_1 future symbols and the $M_2 + \mu$ past symbols. This explains the index range in the definition of $\tilde{\boldsymbol{a}}_k$ and $\bar{\boldsymbol{a}}_k$. Also, note that $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are proportional to the strictly causal and anticausal portions of $\sum_l h_l f_{k-l}$, where the constants of proportionality depend on the quality of $\tilde{a}_k$ and $\bar{a}_k$ through α_1 / E_1 and α_2 / E_2.

5.2.2 Computing the Expected Values

Exploiting symmetries, it is not hard to see from (56)-(59) that E_1, α_1, E_2 and α_2 may be computed by conditioning on $a_k = 1$. In other words, $E_1 = \mathrm{E}[|\tilde{a}_k|^2 | a_k = 1]$, $E_2 = \mathrm{E}[|\bar{a}_k|^2 | a_k = 1]$, $\alpha_1 = \mathrm{E}[\tilde{a}_k | a_k = 1]$, and $\alpha_2 = \mathrm{E}[\bar{a}_k | a_k = 1]$.

Now, assume that λ_k^p is computed from an equivalent AWGN channel with output

$$l_k = a_k + w_k, \tag{62}$$

where w_k is AWGN with variance σ_w^2, assumed to be independent of the transmitted sequence, the actual channel noise and the equalizer output at time k [45]. Assuming a BPSK alphabet, this means that $\lambda_k^p = \gamma_p \, l_k$, where $\gamma_p = 2/\sigma_w^2$ is proportional to the SNR of the equivalent channel that generates λ_k^p. Then, conditioning on $a_k = 1$, $\lambda_k^p \sim \mathcal{N}(\gamma_p, 2\gamma_p)$, so

$$\alpha_1 = \psi_1(\gamma_p), \tag{63}$$

$$E_1 = \psi_2(\gamma_p), \tag{64}$$

where

$$\psi_1(\gamma) = \mathrm{E}[\tanh(u/2)],\ u \sim \mathcal{N}(\gamma, 2\gamma), \tag{65}$$

$$\psi_2(\gamma) = \mathrm{E}[\tanh^2(u/2)],\ u \sim \mathcal{N}(\gamma, 2\gamma). \tag{66}$$

Unfortunately, there are no closed-form formulas for $\psi_1(\gamma)$ and $\psi_2(\gamma)$. However, these are well-behaved functions that may be tabulated or computed by simple numerical algorithms.

Similarly, note that $L_k = \lambda_k^p + \lambda_k$. Now, consider the Gaussian approximation to λ_k in (49) and (50), and let $\gamma_e = 2A^2/\sigma_v^2$ be a parameter that is proportional to the SNR of the equivalent channel that generates λ_k. Then,

$$L_k = (\gamma_p + \gamma_e)\, a_k + \gamma_p w_k + \gamma_e v_k, \tag{67}$$

so that, conditioning on $a_k = 1$, $L_k \sim \mathcal{N}(\gamma_p + \gamma_e, 2(\gamma_p + \gamma_e))$. Thus, using the Gaussian assumptions, E_2 and α_2 are given by

$$\alpha_2 = \psi_1(\gamma_p + \gamma_e) \tag{68}$$

$$E_2 = \psi_2(\gamma_p + \gamma_e). \tag{69}$$

To compute E_1, α_1, E_2 and α_2, the values of γ_p and γ_e need to be estimated. Because $\mathrm{E}[|\lambda_k^p|^2] = \gamma_p^2 + 2\gamma_p$, the ML estimate of γ_p is

$$\hat{\gamma}_p = \sqrt{1 + \frac{1}{N}\sum_{k=0}^{N-1} \left|\lambda_k^p\right|^2} - 1. \tag{70}$$

To determine γ_e, we note that, as shown in Appendix B, $A = \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0$ and $\sigma_v^2 = A(1-A)$. Thus, since $\gamma_e = 2A^2/\sigma_v^2$, we have that

$$\gamma_e = 2\,\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0/(1 - \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0). \tag{71}$$

Note that we need γ_e to compute E_2 and α_2, but we need E_2 and α_2 to compute γ_e. To find both simultaneously, we propose that, given an initial value for γ_e, we compute:

$$\boldsymbol{f} = (\mathbf{H}\mathbf{H}^{\mathrm{T}} - \frac{\alpha_1^2}{E_1}\mathbf{H}_1\mathbf{H}_1^{\mathrm{T}} - \frac{\psi_1^2(\gamma_p+\gamma_e)}{\psi_2(\gamma_p+\gamma_e)}\mathbf{H}_2\mathbf{H}_2^{\mathrm{T}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{h}_0, \tag{72}$$

$$\gamma_e = 2\,\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0/(1-\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0). \tag{73}$$

iteratively, until a stop criterion is met. The converge behavior of this iterative procedure can be studied with the same techniques used in section 3.2 to analyze the scalar-channel estimator. Indeed, for a fixed channel, noise variance and γ_p, the iterative procedure may be seen as a mapping from the value of γ_e at the i-th iteration, γ_e^i, to the value of γ_e at the i+1-th iteration, γ_e^{i+1}. With that in mind, consider Fig. 21, where we plot γ_e^{i+1} as a function of γ_e^i for $\gamma_p = 0$, $\boldsymbol{h} = [0.227\ 0.46\ 0.688\ 0.46\ 0.227]$, $M_1 = 10$, $M_2 = 5$, and $SNR = 10$ dB. Also shown in Fig. 21 is the line $\gamma_e^{i+1} = \gamma_e^i$. Since the iterative procedure converges when $\gamma_e^{i+1} = \gamma_e^i$, Fig. 21 suggests the existence of a single fixed-point for this particular example,

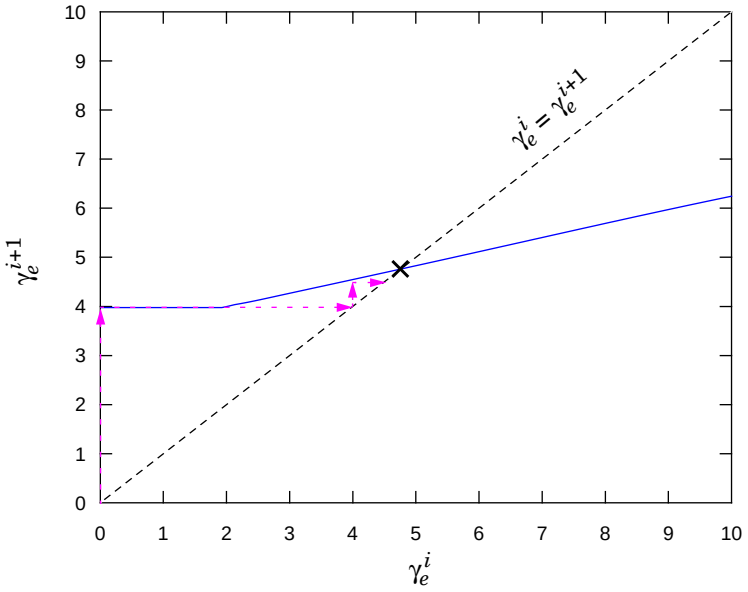

Fig. 21. Graphical analysis of the convergence of (72), (73): estimated value of γ_e^{i+1} as a function of its value at the previous iteration, γ_e^i.

marked by a $\times$. Furthermore, the dynamic behavior of the iterative procedure, when γ_e is initialized to zero, is illustrated in Fig. 21 by the dotted arrows. As seen in Fig. 21, the algorithm is expected to converge after 3 iterations in this case. We have observed the same fast convergence and unimodal behavior in all scenarios we have studied.

To summarize, the SFE coefficients are computed as in the following pseudocode:

```
Estimate γ_p using (70);
Estimate γ_e using the iterative procedure in (72) and (73);
Compute E_1, α_1, E_2 and α_2 using (63), (64), (68), (69);
Compute the SFE coefficients using (52), (53), (54).
```

5.2.3 Special Cases and Approximations

The values of γ_p and γ_e are proportional to the SNR of the equivalent AWGN channels that generate λ_k^p and λ_k, respectively, and hence reflect the quality of these channels. Based on this observation, some interesting conclusions may be drawn from a study of the behavior of $\psi_1(\gamma)$, $\psi_1(\gamma)$ / $\psi_2(\gamma)$ and $\psi_1^2(\gamma)$ / $\psi_2(\gamma)$, which are plotted in Fig. 22 as a function of γ.

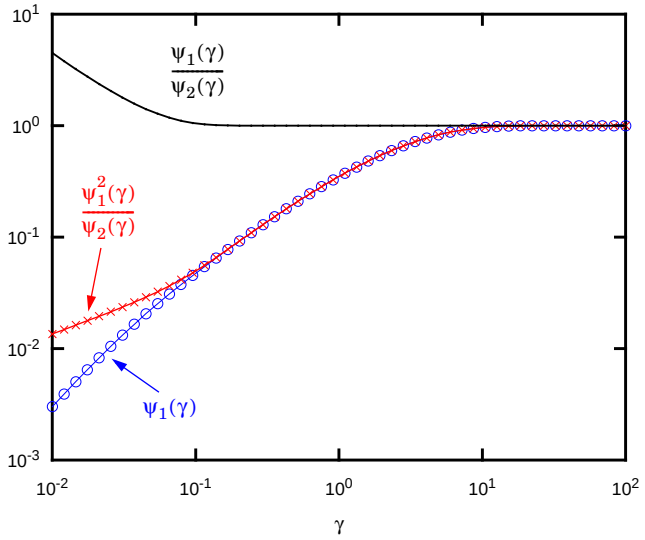

Fig. 22. The behavior of $\psi_1(\gamma)$, $\psi_1(\gamma)/\psi_2(\gamma)$, and $\psi_1^2(\gamma)/\psi_2(\gamma)$.

Consider, for instance, the case in which γ_p or γ_e tends to zero. It can be shown that, as γ tends to zero, the ratio $\psi_1(\gamma) / \psi_2(\gamma)$ tends to infinity, as suggested in Fig. 22. From equations (53) and (54), this implies that the coefficients of the interference cancellers go to infinity as the reliability of their inputs goes to zero. Hence, it seems that the less reliable the symbol estimates, the more we try to cancel their interference. However, this observation is not true. In fact, under the Gaussian assumption it is not hard to show that the outputs of the interference cancellers are zero-mean random variables with variance $(\alpha_1^2 / \mathrm{E}_1)\left\|\mathbf{H}_1^{\mathrm{T}}\boldsymbol{f}\right\|^2$ and $(\alpha_2^2 / \mathrm{E}_2)\left\|\mathbf{H}_2^{\mathrm{T}}\boldsymbol{f}\right\|^2$. It is also possible to show that $\psi_1^2(\gamma) / \psi_2(\gamma)$ tends to zero as the reliability γ tends to zero, as suggested in Fig. 22. Thus, although the filter coefficients grow large, the output of the interference canceller goes to zero in the mean-square sense, and in fact no interference cancellation is done.

Based on Fig. 22, the analysis above, and a careful inspection of (52) – (54), it can be shown that the SFE reduces to well-known equalizers for certain values of γ_p and γ_e.

- In the limit as γ_p and γ_e grow small, we have already shown that no IC is performed. Furthermore, since $\alpha_1^2 / E_1 \to 0$ as $\gamma_p \to 0$ and $\alpha_2^2 / E_2 \to 0$ as $\gamma_e \to 0$, $\boldsymbol{f}$ reduces to a traditional MMSE-LE. Therefore, in this case, the SFE reduces to a conventional linear MMSE equalizer. This is intuitively pleasing, since small values of γ_p and γ_e suggest low levels of reliability, and in this case the receiver is better off not attempting any form of interference cancellation.

- In the limit as $\gamma_p \to 0$ and $\gamma_e \to \infty$, the SFE reduces to a conventional MMSE-DFE. This is intuitive, since small γ_p implies unreliable priors, and hence no cancellation of precursor ISI should be performed. Furthermore, large γ_e implies reliable equal-

izer outputs, in which case postcursor interference can be effectively cancelled using decision-feedback.

- In the limit as $\gamma_p \to \infty$, the SFE reduces to a traditional ISI canceller. This is intuitive because when γ_p is large, the equalizer has access to reliable estimates for all interfering symbols. When the interfering symbols are known, the interference canceller is known to be optimal.

The plot of $\psi_1(\gamma) / \psi_2(\gamma)$ in Fig. 22 also indicates that we could replace α_1 / E_1 and α_2 / E_2 by 1, an approximation that is clearly accurate for $\gamma > 0.1$. To analyze the effects of this approximation for $\gamma < 0.1$, we observe that the feedforward filter $\boldsymbol{f}$, as well as the variance of the output of the interference cancellers, depend on α_1^2 / E_1 and α_2^2 / E_2. In other words, we have to analyze the impact of the approximation on α_1^2 / E_1 and α_2^2 / E_2. However, as suggested in Fig. 22, the difference between $\psi_1(\gamma)$ and $\psi_1^2(\gamma) / \psi_2(\gamma)$ tends to zero as γ tends to zero. Thus, α_1^2 / E_1 and α_2^2 / E_2 are close to α_1 and α_2, respectively, even for unreliable channels. Therefore, approximating α_1 / E_1 and α_2 / E_2 by 1 does not greatly affect the equalizer output. The resulting approximate filter coefficients are thus computed as

$$\boldsymbol{f} = (\mathbf{H}\mathbf{H}^{\mathrm{T}} - \alpha_1 \mathbf{H}_1\mathbf{H}_1^{\mathrm{T}} - \alpha_2 \mathbf{H}_2\mathbf{H}_2^{\mathrm{T}} + \sigma^2\mathbf{I})^{-1} \boldsymbol{h}_0, \tag{74}$$

$$\boldsymbol{g}_1 = \mathbf{H}_1^{\mathrm{T}}\boldsymbol{f}, \tag{75}$$

$$\boldsymbol{g}_2 = \mathbf{H}_2^{\mathrm{T}}\boldsymbol{f}. \tag{76}$$

It is interesting to notice that, under this approximation, the coefficients of the interference cancellers $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are those that would be obtained assuming correct decisions. Thus, the amount of interference cancellation to be performed by the equalizer is controlled

mostly by the amplitude of the soft information, not by the interference cancellation coefficients.

5.3 Performance Analysis

In this section, we present an analysis of the performance of the SFE algorithm, using computer simulations and theoretical results based on the Gaussian approximation. We also compare the SFE to a traditional DFE.

We begin by showing the validity of the Gaussian approximation. To that end, in Fig. 23 we show the estimated probability density function (pdf) of the SFE output, λ_k, based on the transmission of 32,000 bits through the channel $\boldsymbol{h}$ = [0.23 0.42 0.52 0.52 0.42 0.23] at SNR = 20 dB. The equalizer has $M_1 = 20$, $M_2 = 15$, and we do not assume the presence of *a priori* information, *i.e.*, we assume $\lambda_k^p = 0$ for all k. In this figure, we also show the pdf of the LLR at the output of a Gaussian channel with SNR = $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0/(1 - \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0)$, computed using the SFE parameters. As we can see, both pdfs are similar.

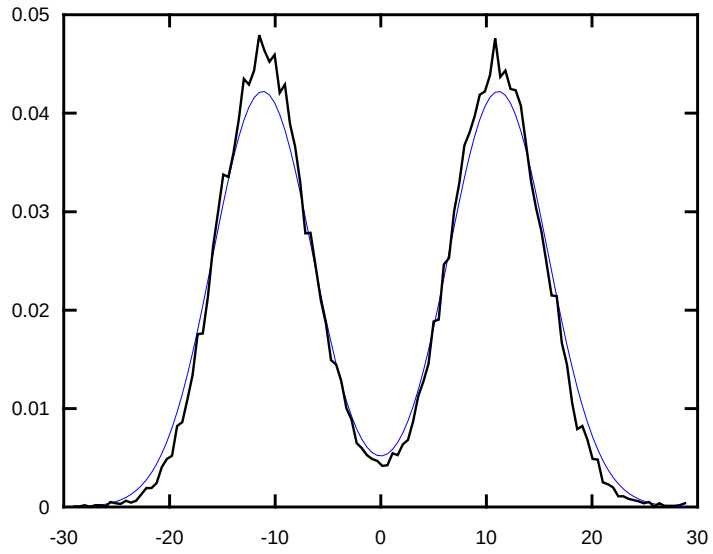

Fig. 23. Estimated pdf of the SFE output, compared to the pdf of the LLR of an AWGN channel.

We may use the fact that the SFE output is similar to the LLR of an AWGN channel to predict the performance of the equalizer. In fact, since the SNR of the equivalent AWGN channel is $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0/(1-\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0)$, the BER at the SFE output should be close to

$$\mathrm{BER}_{\mathrm{SFE}} = \mathrm{Q}\left(\sqrt{\frac{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0}{1-\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0}}\right). \tag{77}$$

To show the accuracy of this computation, we simulate the transmission of 10^7 bits through the channel $\boldsymbol{h} = [0.227\ 0.46\ 0.688\ 0.46\ 0.227]$, using equalizers with $M_1 = 10$, $M_2 = 5$. In Fig. 24, we show the BER performance of the SFE computed through simulations and using (77). As we can see, the theoretical computation in (77) gives a reasonable prediction of the performance of the SFE, especially for low SNR. However, we can also see that equation (77) is normally too optimistic.

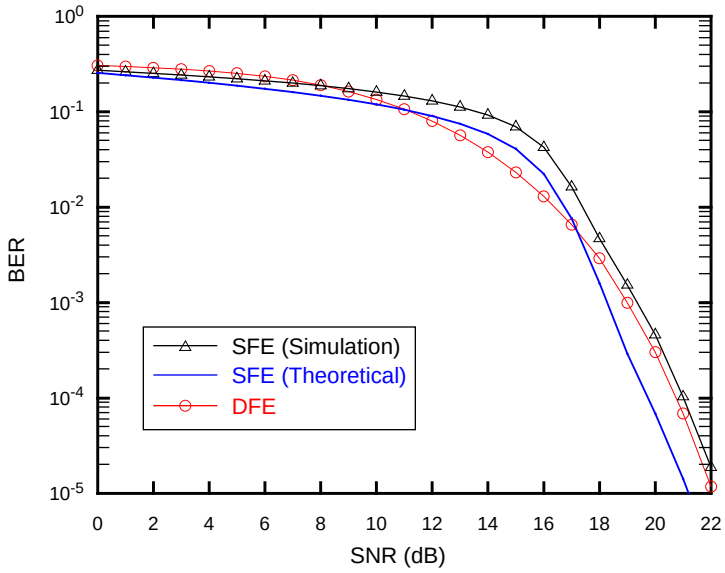

Fig. 24. BER (theoretical and simulation) of an SFE with no *a priori* information. The BER of a DFE is also shown.

In Fig. 24, we also show the BER performance of a DFE. It is interesting to notice that as the SNR increases the performance of the DFE and the SFE become similar. This agrees with the previous analysis, which predicted that the SFE becomes a DFE as the SNR tends to infinity. It is also interesting to notice that the SFE outperforms the DFE for SNR less than 8 dB, while the DFE is better than the SFE for SNR greater than 8 dB. This indicates that, for this channel, there is an SNR threshold above which soft information becomes too tentative, and the equalizer is better off using hard decisions. A similar behavior was observed for other channels. In fact, our simulations indicate that if the SNR at the SFE output, $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0/(1-\boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0)$, is greater than one, then the DFE performs better than the SFE. Unfortunately, a theoretical computation of this SNR threshold could not be determined for a general channel.

5.4 Summary

In this chapter, we proposed the SFE, a low-complexity soft-output equalizer that exploits *a priori* information about the transmitted symbols to perform soft interference cancellation. The SFE achieves a compromise between linear equalization, decision feedback equalization and interference cancellation by choosing the equalizer coefficients according to the quality of the priors and of the equalizer output. Since the SFE exploits *a priori* information, it may replace the BCJR equalizer in applications.

The SFE differs from similar structures [30-35] in two ways. First, it successfully uses soft feedback of the equalizer outputs to improve interference cancellation. In contrast, the decision-feedback structure proposed in [35] performs worse than its linear counterpart.

Also, by assuming a statistical model for the priors, we obtain a time-invariant, linear complexity equalizer, as opposed to the quadratic complexity of the MMSE structures in [32-35].

We showed that the SFE collapses to well-known equalizers in limiting cases where the *a priori* information and the equalizer output are very reliable or very unreliable. We conducted simulations demonstrating the validity of the Gaussian approximation and comparing the performance of the SFE to a DFE. In these simulations, we showed that the SFE outperforms the DFE for low SNR. For high SNR, the performance of both equalizers is similar. For the intermediate SNR range, the DFE outperforms the SFE, suggesting that soft information may be too tentative for this SNR range, and better results are achieved with hard information. This behavior was also observed in other simulations we conducted for different channels, but we could not devise a general strategy to determine when hard information yields better results than soft information.

CHAPTER 6

Turbo Equalization with the SFE

Consider the turbo equalizer shown in Fig. 25, which is repeated here from section 2.2. As discussed in section 2.2, the equalizer in a turbo equalizer computes the LLR of the transmitted symbols, $\boldsymbol{L}^e$, based on the received samples $\boldsymbol{r}$ and the vector of extrinsic information λ^e, and otherwise ignoring the presence of ECC. The vector λ^e is used by the equalizer as *a priori* information on the transmitted symbols; it feeds back information from the decoder to the equalizer, allowing the equalizer to benefit from the code structure. Normally, the equalizer in a turbo equalization scheme is implemented with the BCJR algorithm.

In chapter 5, we proposed the SFE, an equalizer that computes an estimate of LLR of the transmitted symbols, based on the received samples $\boldsymbol{r}$ and the vector of extrinsic information λ^e. Thus, the SFE is well-suited for application in turbo equalization. In this chapter, we study this application of the SFE.

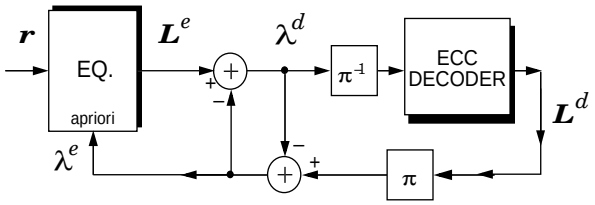

Fig. 25. Turbo equalizer.

6.1 An SFE-Based Turbo Equalizer

An SFE-based turbo equalizer is obtained by using the SFE, depicted in Fig. 20, as the equalizer in the turbo equalization scheme depicted in Fig. 25. The resulting system is depicted in Fig. 26. Note that the SFE coefficients depend on the quality of the *a priori* information. However, in a turbo equalizer, the quality of the *a priori* information changes with iteration. Therefore, the SFE coefficients have to be computed at the beginning of every turbo iteration. In this section, we briefly describe our implementation of the SFE-based turbo equalizer.

First, the computation of the SFE coefficients in a turbo equalization context may be simplified. In the derivation of the SFE, we proposed to compute the parameter γ_e using the iterative procedure described in equations (72) and (73). If we used this strategy in a turbo equalizer, we would have to repeat the iterative procedure for every turbo iteration. However, we have observed that there is no need to this. In fact, the iterative procedure in equations (72) and (73) may be used only in the first turbo iterations. In later turbo

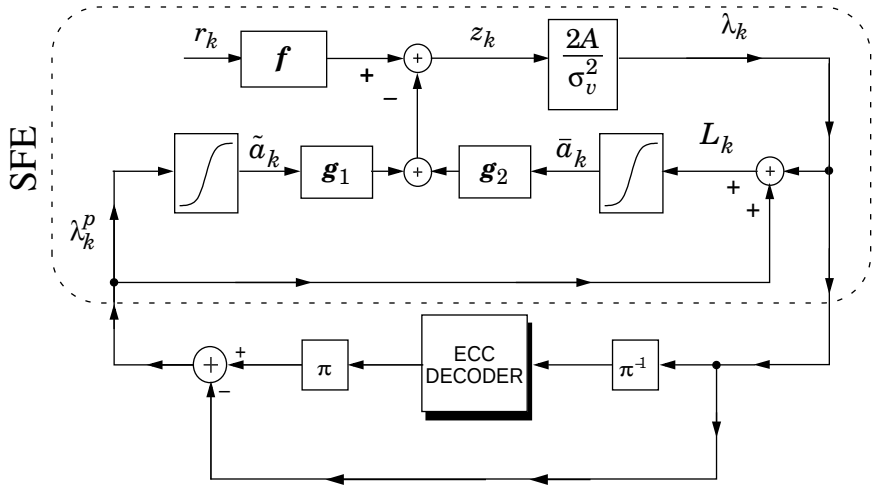

Fig. 26. An SFE-based turbo equalizer.

iterations, we may compute the equalizer coefficients using the value γ_e from the previous turbo iteration. An updated value of γ_e is then computed and passed on to the next turbo iteration. We have observed that no performance loss is incurred if the iterative procedure in equations (72) and (73) is used only in the first turbo iteration.

Also, at the first turbo iteration when $\gamma_p = 0$, we get $E_1 = \alpha_1 = 0$. To avoid the indeterminate α_1 / E_1 in (52) – (54), we artificially set $E_1 = 1$, $\alpha_1 = 0$ for the initial iteration. This is reasonable since, at the first iteration, we do not want to perform any IC based on the *a priori* probabilities. In fact, algorithms based solely on IC often have a problem at the first turbo iteration, when no *a priori* information is available. For example, to solve this problem, [31] proposes that the BCJR algorithm be used for the first iteration. Note that if we are using the approximate SFE coefficients in (74)-(76) the ratio α_1 / E_1 is never computed, so there is no need to artificially set $E_1 = 1$, $\alpha_1 = 0$ for the initial iteration.

Finally, we have observed that a turbo equalizer may benefit from values of γ_p and γ_e more pessimistic than those obtained using (70) and (71). Optimistic values for γ_p and γ_e may cause the equalizer to output values of λ_k that have the wrong sign but a large magnitude, which may cause the turbo equalizer to converge slowly or to a wrong codeword. Performance can be improved if γ_p and γ_e are estimated using the SEM scalar channel estimator analyzed in section 3.2, repeated here for convenience. If $z_k = A a_k + v_k$ is the equalizer output, then, given initial estimates $\hat{A}_0$ and $\hat{\sigma}_0^2$, γ_e is computed iteratively using

$$\hat{A}_i = \frac{1}{L}\sum_{k=0}^{L-1} \tanh(\hat{A}_{i-1} z_k / \hat{\sigma}_{i-1}^2) z_k ,$$

$$\hat{\sigma}_i^2 = \frac{1}{L}\sum_{k=0}^{L-1} \left\| \hat{A}_i \mathrm{sign}(z_k) - z_k \right\|^2 ,$$

$$\hat{\gamma}_e^{(i)} = \frac{2\hat{A}_i^2}{\hat{\sigma}_i^2} , \tag{78}$$

where the index $i > 0$ refers to the turbo iteration. If we replace z_k by λ_k^p in the equations above, we obtain an estimate for γ_p. The initial values $\hat{A}_0$ and $\hat{\sigma}_0^2$ required to compute $\hat{\gamma}_e^{(i)}$ are obtained from the iterative procedure described in equations (72) and (73). For computing $\hat{\gamma}_p^{(i)}$ we set $\hat{\sigma}_0^2 = 2\hat{A}_0$, which reflects our initial approximation that λ_k^p is consistently Gaussian. A consistently Gaussian random variable is a Gaussian random variable whose variance is equal to twice its mean.

6.2 Simulation Results

We present simulation results of turbo equalizers based on several different soft-output equalizers. In all the simulations, the transmitted symbols are encoded a recursive rate-$1/2$ convolutional encoder with parity generator $(1+D^2)/(1+D+D^2)$ followed by an interleaver whose length is equal to the block length. For these simulations, we also assume that the channel parameters are known.

We begin by using the same simulation scenario as [35], in which $K = 2^{15}$ message bits are encoded and transmitted through the channel $\boldsymbol{h}$ = [0.227, 0.46, 0.688, 0.46, 0.227], whose frequency response is shown in Fig. 27. The equalizers use $M_1 = 9$, $M_2 = 5$, and the SNR per message bit is $E_b/N_0 = (E_s/R)/(2\sigma^2)$, where σ^2 is the noise variance, R is

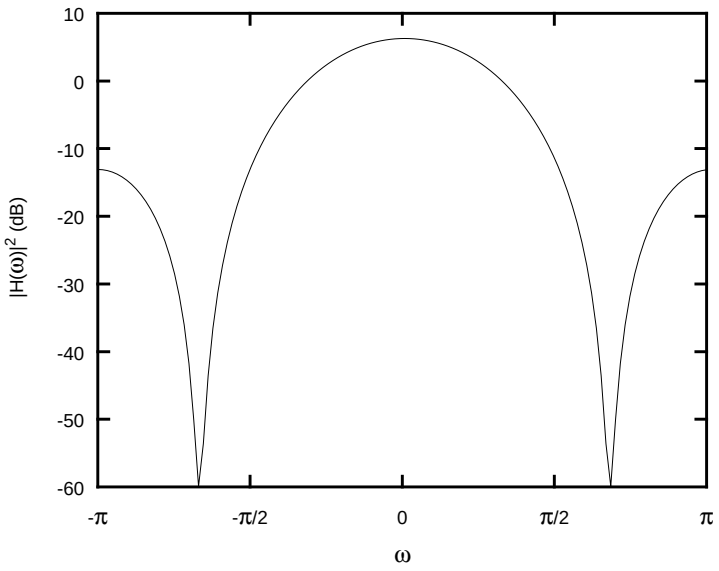

Fig. 27. Frequency response of h = [0.227, 0.46, 0.688, 0.46, 0.227].

the code rate (in this case 1/2) and $E_s = \|\boldsymbol{h}\|^2$ is the symbol energy at the channel output. We have estimated the BER performance as a function of E_b/N_0 of turbo equalizers based on the BCJR algorithm, the SFE, the time-varying equalizer (TVE) from [35], and the switched equalizer (SE) from [35], after 14 iterations of the turbo equalizer. We have also estimated the BER performance of the code in an AWGN channel, which does not introduce ISI. The results, shown in Fig. 28, are averaged over 100 trials. As we can see, the proposed equalizer performs almost as well as the TVE (quadratic complexity), while its complexity is comparable to that of the SE (linear complexity).

In Fig. 28, it is also interesting to see that the performance of all the equalizers eventually approaches that of the coded system in an AWGN channel. Thus, turbo equalization allows for almost perfect ISI removal. In other words, for large enough SNR, turbo systems may operate as if the channel introduced no ISI. Furthermore, in Fig. 28 we show the capacity limit, defined as the minimum E_b/N_0 required for error-free

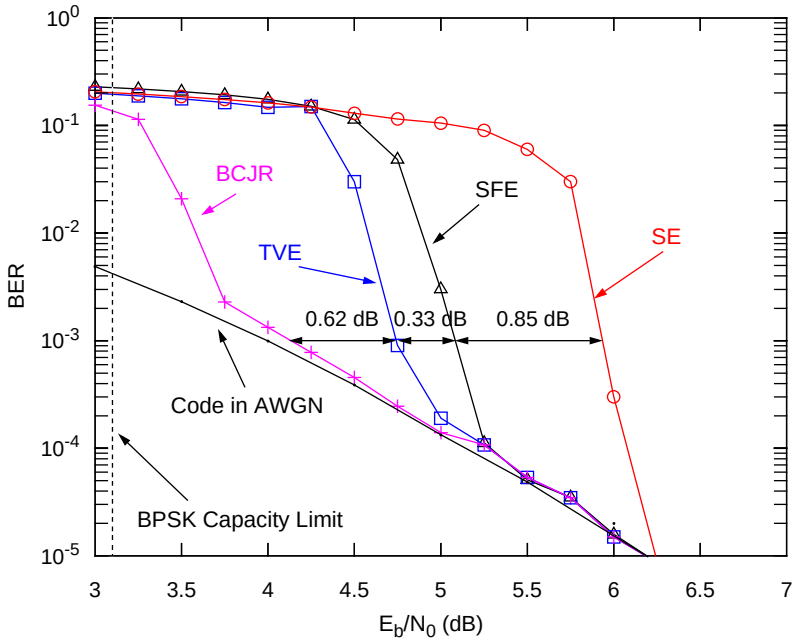

Fig. 28. BER performance for the simulation scenario of [35].

transmission using rate 1/2 codes and BPSK transmission, as predicted by Shannon's theory [46]. This limit was computed using the techniques proposed in [47]. For the channel considered in this simulation, the capacity limit is $E_b/N_0 = 3.06$ dB. As seen in Fig. 28, the gap between the turbo-based systems and the capacity is not wide. Furthermore, the BCJR-based system behaves like an AWGN system for $E_b/N_0 > 3.70$ dB, suggesting that the code, rather than the ISI channel, is responsible for a significant part of this gap. In fact, the gap can be made narrower if better codes are used.

As seen in Fig. 28, the BCJR equalizer yields the best performance among all the equalizers. There is, therefore, a trade-off between the added complexity of the BCJR and its performance gain. To quantify this trade-off, in Fig. 29 we show the number of operations (additions and multiplications) required by the BCJR- and the SFE-based turbo equalizers to achieve a BER of 10^{-3} at a given E_b/N_0. In this figure, we do not take the decoding complexity into account, since this is common to all equalizers. Furthermore, we

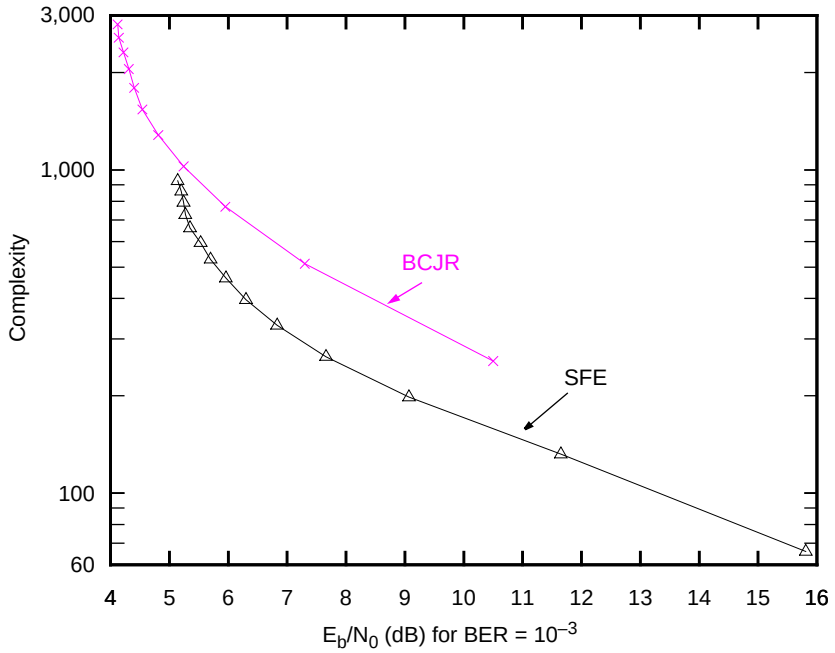

Fig. 29. Complexity-performance trade-off.

do not consider other complicating factors of the BCJR algorithm, such as the significant memory requirements and the constant use of lookup table to compute exponentials and logarithms. Even without taking these factors into account, we see in Fig. 29 that the BCJR-based turbo equalizer requires 1.8 times as many operations as the SFE-based turbo equalizer to achieve a BER of 10^{-3} at $E_b/N_0 = 7$ dB. Likewise, if we are limited to 300 operations, the SFE-based system can operate at an E_b/N_0 2.7 dB less than that made possible by the BCJR-based system.

The performance gap between the different techniques is a strong function of the channel. To see this, we simulate the transmission of $N = 2^{11}$ encoded bits through $\boldsymbol{h} = [0.23, 0.42, 0.52, 0.52, 0.42, 0.23]$, whose frequency response is shown in Fig. 30. This is the 6-tap channel that causes maximum performance degradation for the ML sequence detector when compared to the matched filter bound [48]. We used $M_1 = 15$ and $M_2 = 10$.

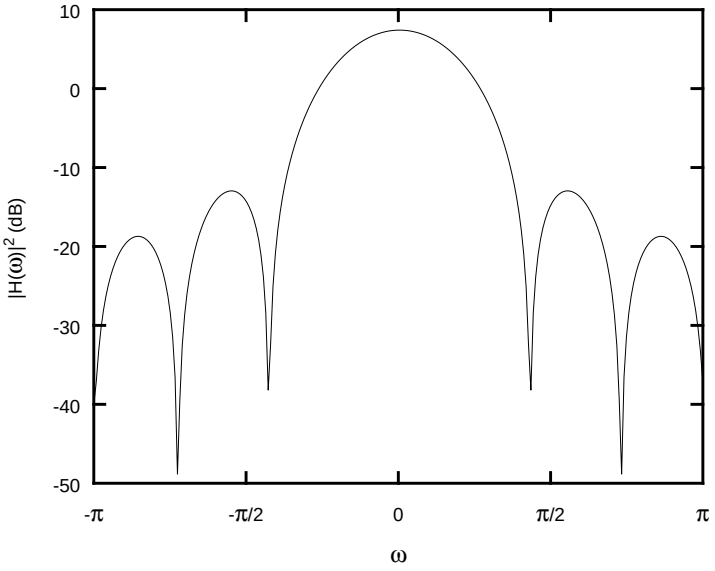

Fig. 30. Frequency response of h = [0.23, 0.42, 0.52, 0.52, 0.42, 0.23].

For each value of E_b/N_0 and every 30 codewords, we checked the total number of words detected in error. If this number was greater than 100, we would stop running the simulation for that value of E_b/N_0. A maximum of 1000 codewords was transmitted for each E_b/N_0.

The performance of the turbo equalizers based on BCJR, DAE, SE and the SFE for the simulation scenario described above is shown in Fig. 31, where we plot the BER versus E_b/N_0 for the turbo equalizers. The maximum number of iterations shown for each scheme is that after which the equalizers stopped improving. It is interesting to notice that, for the DAE, error propagation is a problem for $E_b/N_0 < 10$ dB, as evidenced by its poor performance in this SNR range. After this value, the performance improves rapidly with increasing E_b/N_0. It is important to point out that the first turbo iteration for this algorithm uses a BCJR equalizer, which precludes its application to channels with long memory. We can also see in Fig. 31 that the SFE is around 2.6 dB better than the SE

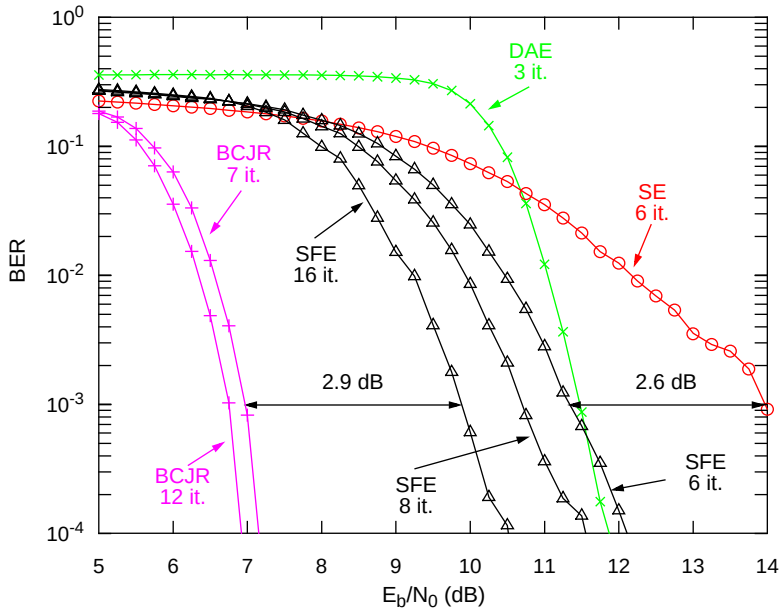

Fig. 31. BER performance of some turbo equalizers for h = [0.23, 0.42, 0.52, 0.52, 0.42, 0.23]. The BPSK capacity limit for this scenario is E_b/N_0 = 4.2 dB.

equalizer for the same number of iterations and a BER of 10^{-3}. However, performance cannot be further improved with the SE equalizer. On the other hand, with the SFE a gain of 0.65 dB is possible with 2 extra iterations, and a 1.4 dB gain is possible with 10 more iterations. One possible explanation for this performance gap is that, as seen in Fig. 30, the channel used in this simulation introduces severe ISI. For such channels, decision feedback structures such as the proposed algorithm tend to perform better than linear filters.

The BPSK capacity limit for the simulation scenario used to generate Fig. 31, computed using the techniques proposed in [47], is E_b/N_0 = 4.2 dB. Therefore, the gap to capacity for this particular channel is larger than the gap in Fig. 28, thus indicating that the gap to capacity depends on the channel characteristics. It should be mentioned that the turbo systems considered in Fig. 31 do approach the performance of an ISI-free system; however, this happens at a low BER, for which we have no results.

The channels we have considered so far have a fairly short impulse response. Using a BCJR equalizer for these channels is a feasible option, and there is a trade-off between some extra computational burden and some performance gain. However, in channels with very long impulse responses the complexity of the BCJR equalizer is prohibitive, and using this equalizer is no longer an option. To obtain the gains of turbo equalization in channels with long impulse responses, low-complexity equalizers have to be used. Consider, for instance, the microwave channel of [49]. For this example, we focus on the 44-tap section of this channel, corresponding to samples 98 through 141. Furthermore, since we are using a BPSK modulation, we use only the real part of the channel. The resulting impulse response is shown in Fig. 32, and the frequency response is shown in Fig. 33. For such a long channel, the complexity of BCJR is roughly 2^{47} additions and multiplications per symbol per iteration, and even quadratic-complexity equalizers such as

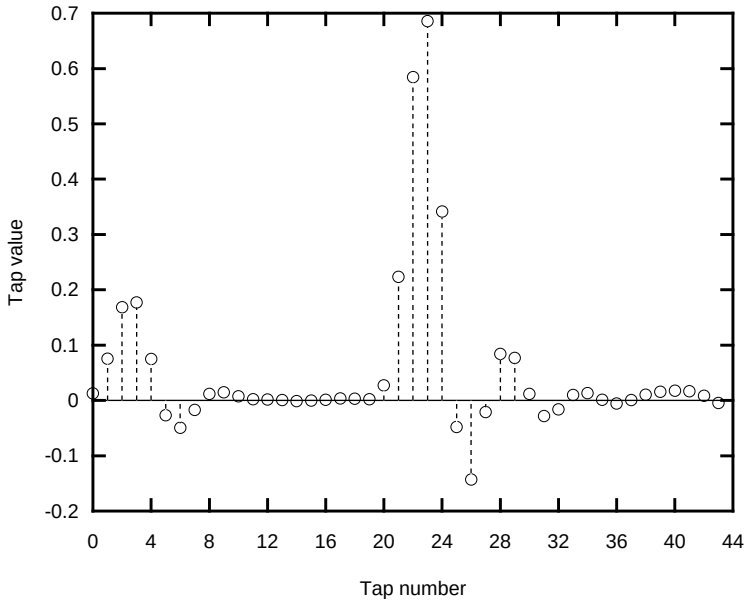

Fig. 32. Impulse response of microwave channel of [49].

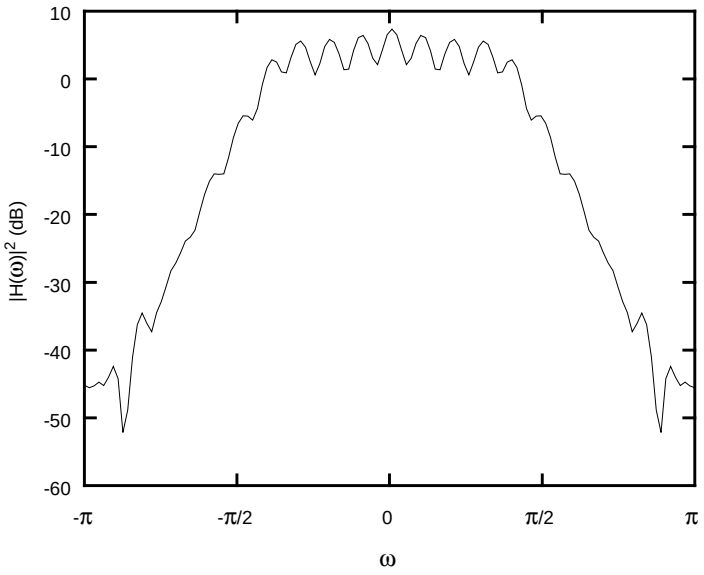

Fig. 33. Frequency response of microwave channel of [49].

the TVE are too complex. In cases like this, linear complexity equalizers are the only feasible choice. Therefore, to use a turbo equalizer for this channel, we need to use either the SFE or the SE.

To determine the performance of the SFE- and SE-based turbo equalizers for the microwave channel, we simulate the transmission of $N = 2^{11}$ encoded bits through this channel. We used equalizers with $M_1 = 40$ and $M_2 = 20$. A maximum of 1000 codewords was transmitted for each E_b/N_0. For each value of E_b/N_0 and every 30 codewords, we checked the total number of words detected in error. If this number was greater than 100, we would stop running the simulation for that value of E_b/N_0. In Fig. 34, we plot performance of the turbo equalizers based on the SE and the SFE, in terms of BER versus E_b/N_0. In this figure, the gains of turbo equalization are clear, as evidenced by the performance gap between the first and 8-th iteration of both turbo equalizers. We can also see that the SFE-based turbo equalizer outperforms the SE-based turbo equalizer, with a gap of 1.5 dB for a BER of 10^{-3}. The capacity limit for the microwave channel is also

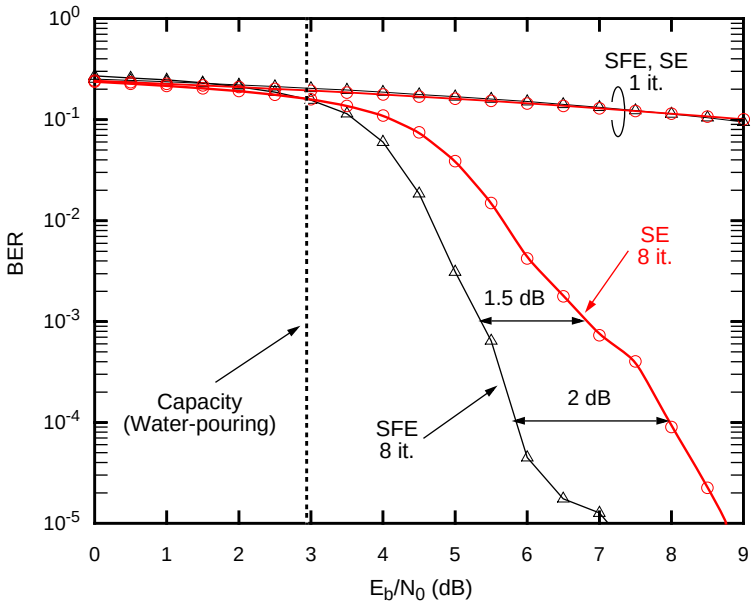

Fig. 34. BER performance of the SFE- and SE-based turbo equalizers for the microwave channel.

shown in Fig. 34. However, since the complexity of the technique proposed in [47] grows exponentially with channel memory, computing the BPSK capacity is not feasible for this channel. Therefore, we show the power-constrained capacity, computed using water-pouring [46]. As we can see, the SFE-based system has a gap to capacity of around 3 dB, which can be made narrower if better codes are used.

6.3 The EXIT Chart

In this section, we describe the extrinsic information transfer (EXIT) chart, a design tool for iterative systems such as turbo equalizers. We also compare the EXIT charts of different equalizers.

The EXIT charts were originally proposed in [45] for the analysis of parallel-concatenated turbo codes, but they can also be used for turbo equalization. The idea behind these charts is that the equalizer or the decoder in an iterative detector can be seen

as a block that maps extrinsic information at its input, λ_{in}, to extrinsic information at its output, λ_{out}. Furthermore, λ_{in} can be characterized by a single parameter, the mutual information between λ_{in} and the transmitted symbols, I_{in}. For a binary alphabet, we get

$$I_{in} = \sum_{a \in \{\pm 1\}} \int p_{in}(\lambda|a) \log_2 \frac{2p_{in}(\lambda|a)}{p_{in}(\lambda|1) + p_{in}(\lambda|-1)} d\lambda, \tag{79}$$

where $p_{in}(\lambda \mid a)$ is the pdf of λ_{in} given that a was transmitted. The mutual information between λ_{out} and the input sequence, I_{out}, may be similarly defined. Therefore, decoders and equalizers in an iterative receiver may be seen as functions T_e and T_d, respectively, that map I_{in} to I_{out}, as depicted in Fig. 35 for a turbo equalizer. The EXIT chart is a plot of T_e and/or T_d.

For the equalizer, the mapping from I_{in} to I_{out} is estimated for a specific channel and a specific SNR. We assume that λ_{in} is generated from an AWGN channel whose noise component is independent of the channel noise and the transmitted symbols, as was done in (62). Under this assumption, λ_{in} can be generated directly, without regard for the decoder. To compute I_{out}, a long sequence of channel outputs and *a priori* information is generated, and the equalizer is used to produce a sequence of values λ_{out}. The pdfs $p_{out}(\lambda \mid 1)$ and $p_{out}(\lambda \mid -1)$ are then estimated based on histograms of the equalizer output,

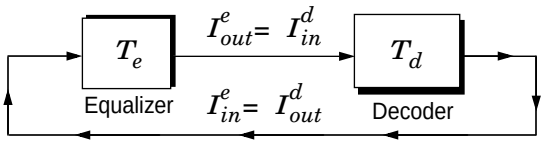

Fig. 35. View of turbo equalization as an iterative application of mappings.

and I_{out} is obtained from a numerical computation of the integral in (79). The mapping for the decoder is estimated in a similar fashion. The only difference is that for the decoder there is no channel output, only extrinsic information.

The EXIT chart can be used to graphically predict the behavior of the iterative algorithm. To see this, consider Fig. 36, where we plot the EXIT charts (I^e_{out} as a function of I^e_{in}) for the BCJR equalizer, the SFE and the SE. The charts were obtained for the channel given by $\boldsymbol{h}$ = [0.227, 0.46, 0.688, 0.46, 0.227], and for an E_b/N_0 = 5.1 dB, using 10^7 transmitted symbols. We also show I^d_{in} (which is equal to I^e_{out}) as a function of I^d_{out} (which is equal to I^e_{in}) for the recursive rate-$1/2$ convolutional encoder with parity generator $(1+ D^2)/(1 + D + D^2)$. Note that $I^d_{in} = T^{-1}_d(I^d_{out})$. Thus, the plot for the decoder can be obtained by switching the abscissa and the ordinate in the decoder EXIT chart. The iterative procedure for the BCJR equalizer is represented by the arrows in Fig. 36.

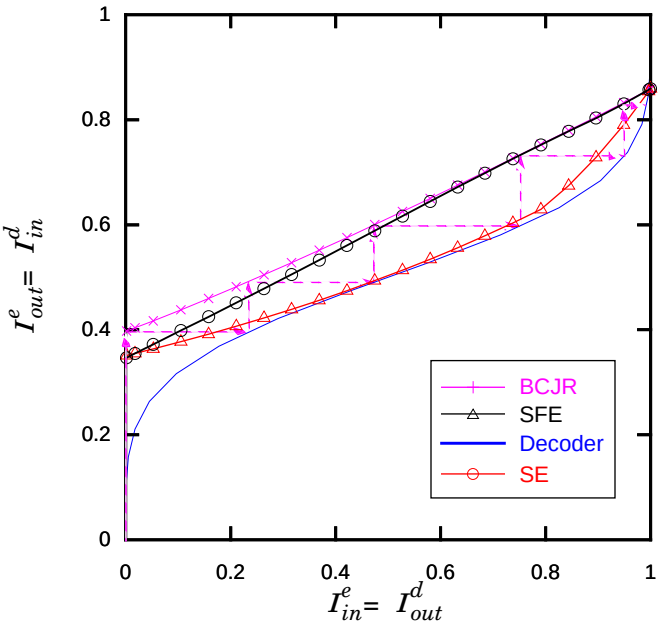

Fig. 36. The EXIT charts for the BCJR equalizer, the SFE and the SE at E_b/N_0 = 5.1 dB and for $\boldsymbol{h}$ = [0.227, 0.46, 0.688, 0.46, 0.227]. The flipped decoder chart is also shown.

Initially, the equalizer has no extrinsic information, so that $I_{in}^{e} = 0$, so it produces an output with $I_{out}^{e} = T_e(0)$, represented by the first vertical arrow. The decoder then produces an output with $I_{in}^{e} = T_d(I_{out}^{e})$, a value that can be found graphically through the first horizontal arrow. (Remember that for the decoder we plot $I_{out}^{e} = T_d^{-1}(I_{in}^{e})$.) With this new value of I_{in}^{e}, the equalizer produces a new value $I_{out}^{e} = T_e(I_{in}^{e})$, found with the second vertical arrow. The iterative procedure progresses in the same fashion in future iterations.

As seen in Fig. 36, for both the BCJR equalizer and the SFE the mutual information tends to a large value (close to 1) as the iterations progress, which implies a small BER [45]. For the SE, however, the mutual information stops increasing at a small value, which implies a large BER. Furthermore, we see that the BCJR-based turbo equalizer is expected to converge in 5 iterations, while the SFE-based turbo equalizer is expected to converge in 6 iterations. Finally, we see in Fig. 36 that for no extrinsic information ($I_{in}^{e} = 0$) the SFE produces the same I_{out}^{e} as the SE. However, when $I_{in}^{e} = 0$ the SE chooses the MMSE-LE. These two observations indicate that the SFE performs like an MMSE-LE. This is intuitively pleasing, since in the absence of *a priori* information and with unreliable equalizer outputs (caused by the low SNR), the SFE should indeed be similar to an MMSE-LE. We can also see in Fig. 36 that, as the reliability of the extrinsic information increases, the SFE, the SE and the BCJR equalizer produce the same I_{out}^{e}. This agrees with the analysis in section 5.2.3 that showed that when the reliability of the extrinsic information tends to infinity, the SFE tends to an MF with IC, which is the ML receiver in this case. (Remember that the SE is an MF when the *a priori* information is reliable.)

The EXIT chart may be used to determine the threshold SNR for a turbo equalizer, defined as the SNR above which the turbo equalizer converges to a small BER, and below which the turbo equalizer does not converge to a small BER. From the iterative procedure described above, it is clear that the turbo equalizer will converge to a small BER if the EXIT chart for the equalizer only intersects the inverted decoder chart at a high value of the mutual information. As seen in Fig. 36, the plots for the SE and the decoder intersect at a small value of I_{in}^{e}, yielding a large BER. However, if we increase the SNR, the curve for the SE will move up, so that the curves only intersect at a high mutual information, resulting in a small BER. Thus, the value of SNR that makes the EXIT chart for the equalizer touch that of the decoder at a single point for a small mutual information is the SNR threshold for that equalizer. Using this procedure, we determine the threshold for the BCJR equalizer, the SFE and the SE. The resulting thresholds for the channel $\boldsymbol{h}$ = [0.227, 0.46, 0.688, 0.46, 0.227] are shown in Table 2.

The EXIT chart has another interesting application, stemming from the fact that they may be generated for the equalizer and decoder independently. This allows for different combinations of coding and equalization techniques to be compared directly, without the need to simulate a turbo equalizer for each combination. This property of EXIT charts makes it useful for designing codes for turbo equalization [50].

Table 2: Threshold SNR for Some Equalizers

Equalizer type	SNR Threshold (dB)
BCJR	3.4
SFE	4.5
SE	5.3

6.4 Summary

In this chapter, we studied the application of the SFE for turbo equalization. We first proposed some modifications to the computation of the SFE coefficients that reduce the computational complexity of the system and improve its performance. We then showed, through simulations, that an SFE-based turbo equalizer may perform within 1 dB of a BCJR-based turbo equalizer, which has exponential complexity in the memory of the channel, and within 0.4 dB of a TVE-based turbo equalizer, which has quadratic complexity in the memory of the channel. We showed that the SFE-based turbo equalizer consistently outperforms turbo equalizers based on other linear complexity equalizers.

We have also discussed EXIT charts, a tool for the design of iterative systems. We have provided the EXIT charts of the SFE and compared them with the charts for the BCJR equalizer and the SE. As expected, the SFE is seen to perform between the SE and the BCJR equalizer.

CHAPTER 7

ECC-Aware Blind Channel Estimation

As we have discussed so far, the principle behind the success of iterative techniques is that of exchanging information between blocks so that at every iteration each block uses information from other blocks to improve the reliability of its output. Within that philosophy, in this chapter we combine turbo equalizers and iterative channel estimators in a single system in which channel estimation is based on the decoder output instead of the equalizer output. This way the channel estimator benefits from the code structure, which makes the decoder output more reliable than that of the equalizer, resulting in a blind iterative ECC-aware channel estimator. In a way, this is not very different from the EW channel estimator of chapter 4, since the combination of an equalizer and a decoder can be seen as a symbol estimator.

7.1 ECC-Aware Blind Estimation of a Scalar Channel

In this section, we explain the idea behind ECC-aware channel estimation in simple terms, by considering a scalar channel as shown in Fig. 37. Given initial estimates $\hat{A}_0$ and $\hat{\sigma}_0^2$, the ECC-ignorant approach to estimating the channel gain A and the noise variance σ^2 computes:

$$\hat{A}_{i+1} = \frac{1}{N}\sum_{k=0}^{N-1} r_k \tanh(\hat{A}_i r_k/\hat{\sigma}_i^2), \tag{80}$$

$$\hat{\sigma}_{i+1}^2 = \frac{1}{N}\sum_{k=0}^{N-1} |r_k - \text{sign}(r_k)\ \hat{A}_{i+1}|^2. \tag{81}$$

The idea behind these equations is that $2\ \hat{A}_i r_k/\hat{\sigma}_i^2$ is an estimate of the LLR of the channel input a_k, which is equal to $2Ar_k / \sigma^2$ if only the channel is taken into account. We may, however, obtain a better estimate for this LLR by taking the code into account. This can be done by considering the output of the BCJR decoder. In this case, we may estimate

$$\hat{A}_{i+1} = \frac{1}{N}\sum_{k=0}^{N-1} r_k \tanh(\lambda_k/2), \tag{82}$$

$$\hat{\sigma}_{i+1}^2 = \frac{1}{N}\sum_{k=0}^{N-1} |r_k - \text{sign}(\lambda_k)\ \hat{A}_{i+1}|^2. \tag{83}$$

7.2 ECC-Aware Blind Estimation of an ISI Channel

In this section, we propose an ECC-aware blind channel estimator for a general channel. The application of the principle discussed in the previous section to a channel that introduces ISI is not straightforward. After all, when discussing turbo equalization, we saw that it is hard to obtain the APP on the transmitted symbols while taking the code structure into account. However, we also saw that turbo equalizers may produce an approximation to this APP. In fact, one important aspect of turbo equalizers is that they

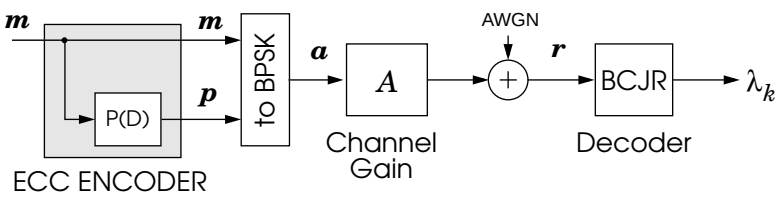

Fig. 37. A simple encoded scalar channel.

provide soft estimates of the transmitted sequence that benefit from the ECC code structure, and are much more reliable than the estimates provided by an equalizer alone. Its seems natural that using this information for channel estimation should provide better results than using ECC-ignorant symbol estimates, as is done in Fig. 5. Thus, we propose the channel estimator of Fig. 38, in which the symbol estimator in Fig. 5 is replaced by the turbo equalizer of Fig. 3.

The proposed estimator of Fig. 38 iterates between three blocks: a channel estimator, a soft-output equalizer, and a soft-output ECC decoder. A receiver would have to perform these functions anyway, so their presence alone does not imply any added complexity; the only added complexity is due to the fact that these functions are performed multiple times as the algorithm iterates.

It is instructive to compare the proposed estimator with a conventional receiver that performs channel estimation just once, then uses these estimates in a turbo equalizer. The proposed estimator can be derived from this receiver by making just one modification:

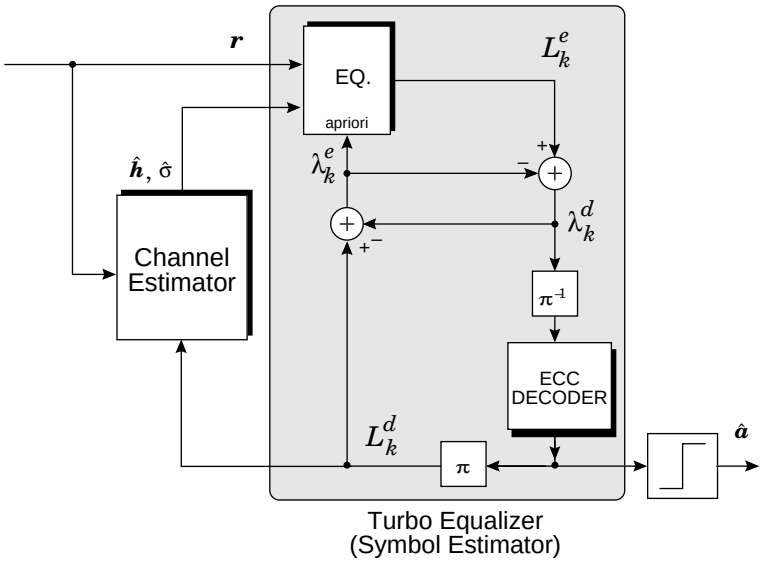

Fig. 38. Integrating channel estimation with turbo equalization.

rather than using the initial channel estimates for every turbo iteration, the proposed receiver occasionally improves the channel estimates based on tentative soft decisions. Specifically, every J-th iteration of the turbo equalizer, the soft-symbol estimates produced by the ECC decoder are used by the EW algorithm to produce better channel estimates, which are then used for the next J iterations. Key to the good performance is the fact that the *a priori* information for the turbo equalizer is not initialized to zero after J iterations of the turbo equalizer. Instead, extrinsic information from the last instance is used as the initial *a priori* information in the next one. The choice of J is a design parameter that can affect convergence speed, steady-state behavior, and overall complexity. Because of the low complexity of the channel estimator relative to the complexity of the equalizer and ECC decoder, we have found empirically that $J = 1$ is a reasonable choice. With this choice, each time the ECC decoder passes extrinsic information to the equalizer, the channel estimates are simultaneously improved. This is only marginally more complex than a conventional receiver that uses turbo equalization, but the performance improvement that results can be significant.

One complicating factor for ECC-aware blind channel estimators is the presence of the interleaver. As is well-known [7], the output of a blind equalizer, or of an equalizer based on a blind estimator, is a delayed version of the channel input, and this delay cannot be compensated for blindly. However, if a delayed sequence is fed to the deinterleaver in Fig. 38, the decoder input will be practically independent of the encoder output. In this case, the decoder output will also be practically independent of the channel input, so that the channel estimates are almost zero. Thus, if a delay is present the blind ECC-aware channel estimator fails. As seen in chapter 4, the EW algorithm exploits the knowledge of

the channel memory to estimate this delay. In this chapter, we assume that the channel memory is known, so the delay problem is not considered. This memory has to be estimated in practice. If this estimate is not accurate, the delay will have to be estimated using some nonblind technique.

7.3 Simulation Results

In this section, we present simulation results of a system with ISI. We compare the performance of the blind ECC-aware system to one with channel knowledge. We also compare the performance of the ECC-aware and ECC-ignorant estimators.

We begin by comparing the performance of the blind scheme to a turbo equalization scheme with channel knowledge. The channel is given by $\boldsymbol{h}$ = [1 0.9 0.8], and the BCJR algorithm is used to provide estimates of the LLR of the channel input $\boldsymbol{a}$. The generator polynomial for the channel encoder is $P(D) = (1 + D^2) / (1 + D + D^2)$. The channel and noise variance estimates are initialized to what we call an *impulsive initialization*: $\hat{\sigma}_0^2 = 1/(2N)\sum_{k=0}^{N-1}|r_k|^2$ and $\hat{\boldsymbol{h}}_0 = [\hat{\sigma}_0, 0, \ldots 0]$. This initializes the channel to a single-tap gain, and initializes the noise to an initial SNR of 0 dB, while keeping the values of $\hat{\boldsymbol{h}}_0$ and $\hat{\sigma}_0$ consistent with the received energy. Fig. 39 shows the BER versus E_b/N_0 for the blind scheme and for a turbo equalizer with channel knowledge. The results are an average of the transmission of 320 blocks of 2048 message bits each. The advantages of ECC-aware channel estimation are clear. We can see that the large gap between the blind and non-blind schemes at one iteration is reduced to virtually nothing at seven iterations.

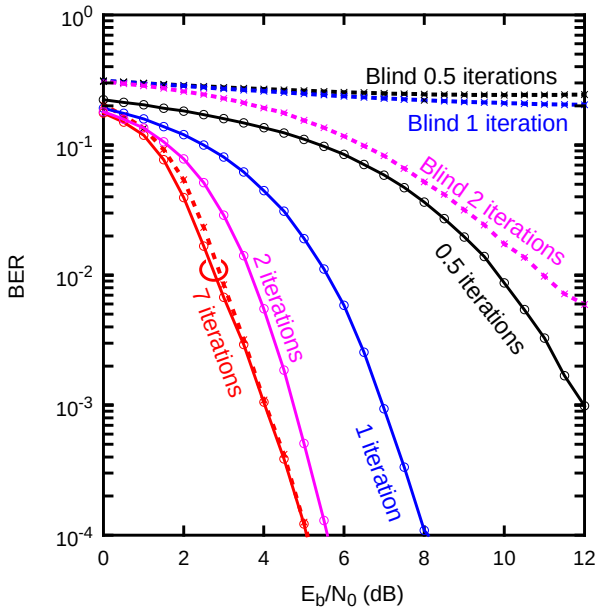

Fig. 39. Comparison between ECC-aware blind channel estimation and channel knowledge in turbo equalization.

It is also worthwhile to compare the performance of the ECC-aware and the ECC-ignorant channel estimates. To that end, we run the same experiment as before. However, for the ECC-ignorant estimator, the channel is estimated with the equalizer output, and no ECC decoding is performed. In Fig. 40, we plot the estimation error *versus* E_b/N_0 for both these estimators, where the solid lines represent the errors after nine iterations and the dotted lines represent the errors for previous iterations. We can see that the ECC-aware estimator yields a gain of about 4 dB when compared to the ECC-ignorant estimator.

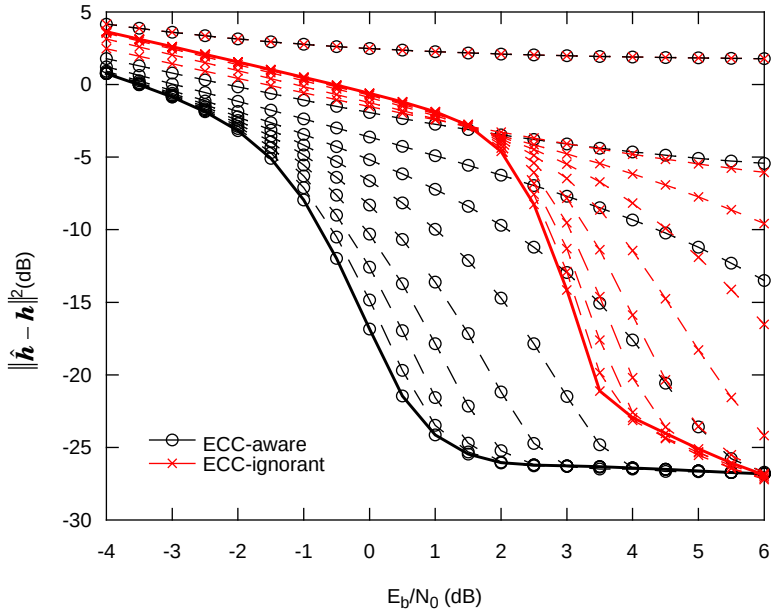

Fig. 40. Comparison between ECC-aware and ECC-ignorant blind channel estimation.

7.4 Study of Convergence

To study the convergence of the ECC-aware channel estimator, and to compare it to the ECC-ignorant estimator, we will use both these algorithms to estimate randomly generated channels. We declare a channel to be successfully estimated if the turbo equalizer based on these estimates correctly detects the transmitted codeword, without any errors.

By defining a successful channel estimate as that which yields no errors, an attempt to estimate a channel may fail not because of an intrinsic convergence problem with the estimation algorithm, but rather because of misconvergence of the turbo equalizer. In fact, it is known that there are "bad" sequences for a turbo system, which cause many detection errors. For example, in the transmission of say 1,000 information bits with a BER of 10^{-3}

using turbo detection, one is not to expect one bit error at every block, but rather one block with around 200 bit errors every 200 blocks. In other words, some of the failures in the random channel experiment are due to these "bad" channel outputs, not to an inherent convergence problem with the blind channel estimator. In that sense, we must compare the success of the blind algorithm with the success of the algorithm with channel knowledge.

We based our analysis on the random channel experiment, in which a set of 400 information bits were generated and encoded using a rate 1/4 serially concatenated turbo code consisting of two recursive systematic convolutional codes with generator polynomial $P(D) = (1 + D) / (1 + D + D^2)$, and an interleaver between the encoders. The resulting encoded sequence was then interleaved and transmitted through a 5-tap ISI channel. We have conducted 1,000 such experiments, in which the channel, the transmitted and noise sequences, and the interleavers were generated randomly. The channel coefficients in particular are generated according to $\boldsymbol{h} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. The signal to noise ratio was kept constant at $E_b/N_0 = 2$ dB. The noise variance and channel estimates were initialized with an impulsive initialization.

For the ECC-ignorant case, we obviously cannot expect that the WER after equalization be zero. However, we may use the estimates provided by the ECC-ignorant estimator in a turbo equalization setting. In this case, the channel estimates are not updated as the turbo equalizer iterates. We may then consider an ECC-ignorant channel estimate to be successful if the turbo equalizer based on this estimate produces zero errors. Thus, the success of the ECC-ignorant estimator was measured by running a turbo equalizer that uses ECC-ignorant estimates, which were obtained after 30 iterations of the ECC-ignorant channel estimator. In Fig. 41 we plot the word-error rate (WER) *versus* iteration for the

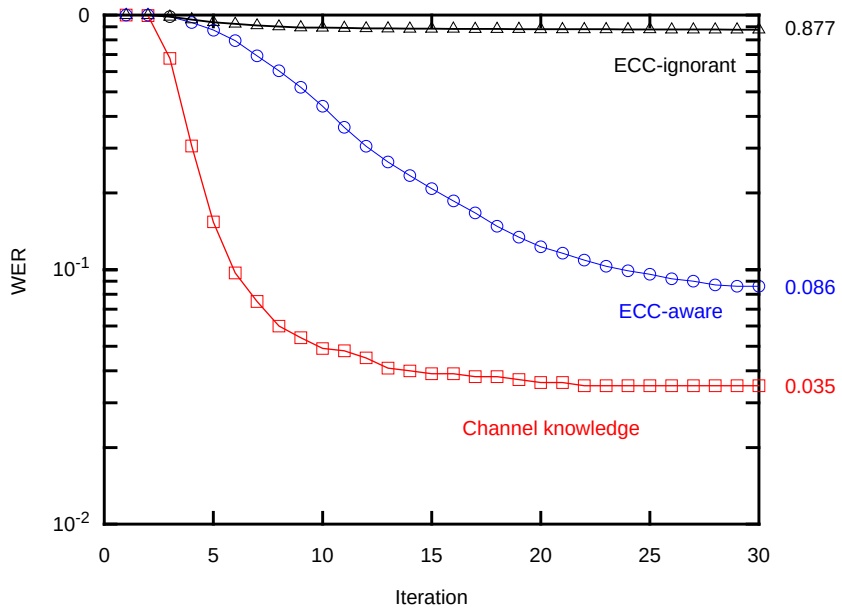

Fig. 41. Word-error rate (WER) across different channels.

ECC-aware and ECC-ignorant blind schemes, and for a turbo equalizer with channel knowledge, henceforth referred to simply as turbo equalizer. We say that the algorithm produced a word-error at a given iteration if its output did not coincide with the transmitted codeword. This plot highlights the dramatic need for ECC-aware channel estimation. We see that the ECC-ignorant estimates are very poor, and that the code is not able to compensate for this estimation error, yielding a high WER. This plot also shows the fact that the WER of the blind scheme is not 0% not only because the blind channel estimator may fail, but also because the turbo detection may fail. In fact, the turbo equalizer presents a WER of 3.5%, when compared to a WER of 8.6% for the blind scheme. What is even more interesting is that the blind scheme does not fail whenever the turbo equalizer fails. Even though this is true for 34 out of the 35 channels in which the turbo equalizer fails, for one run of this experiment the blind scheme was able to correctly detect the transmitted codeword while the turbo equalizer was not.

7.5 Initialization

One obvious issue that arises from the considerations in the previous section is that of initialization. One may wonder whether some failures of the blind algorithm could have been avoided if the channel estimates were initialized to a value closer to the actual channel than the impulsive initialization. Another possible advantage of good initialization is faster convergence, hence lower complexity. As seen in Fig. 41, the blind scheme may take longer to converge than the turbo equalizer with channel knowledge. Thus there is room for improving speed of convergence, and this may conceivably be achieved with a smart initialization.

In this section, we explore the use of the cumulant matrix subspace (CMS) algorithm [51] to initialize the ECC-aware blind estimator. This algorithm belongs to a class of blind channel estimators that exploits the higher-order statistics (HOS) of the received signal to provide a generally simple and closed form channel estimate. These algorithms are generally used to initialize other HOS blind algorithms that provide better channel estimates, but are more complex and more prone to misconvergence, requiring good initialization. Given the intended use of CMS, it seems natural to use this algorithm to initialize iterative channel estimators.

Before studying the impact of a CMS initialization in ECC-aware estimation, we again show more evidence of the benefits of exploiting the code structure for channel estimation. We do that with the same experiment used in generating Fig. 40, *i.e.*, 320 blocks of 2048 bits each are encoded with a rate 1/2 recursive systematic convolutional code with generator polynomial $P(D) = (1 + D^2) / (1 + D + D^2)$, interleaved and transmitted through a channel given by $\boldsymbol{h} = [1\ 0.9\ 0.8]$. The BCJR algorithm is used to provide estimates of the

LLR of the channel input $\boldsymbol{a}$. This time, the channel and noise variance estimates are initialized with estimates provided by the CMS algorithm. In Fig. 42 we show the estimation error of the CMS algorithm and of the ECC-aware estimator after 9 iterations (solid lines), as well as for the iterations in-between (dotted lines). It is clear that using ECC-aware estimation after the CMS algorithm greatly improves the estimates, with a significant gain of 7 dB of E_b/N_0 for estimation errors of –20 dB .

To study the impact of CMS initialization on the convergence of the ECC-aware estimator, we repeat the random channel experiment used to generate Fig. 41, in which in which a set of 400 information bits were encoded using a rate 1/4 serial concatenated turbo code consisting of two recursive systematic convolutional codes with generator polynomial $P(D) = (1 + D) / (1 + D + D^2)$, and transmitted through a channel generated according to $\boldsymbol{h} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, with an E_b/N_0 of 2 dB. We test three different initialization strategies, the impulsive initialization, initialization with CMS and a third strategy that

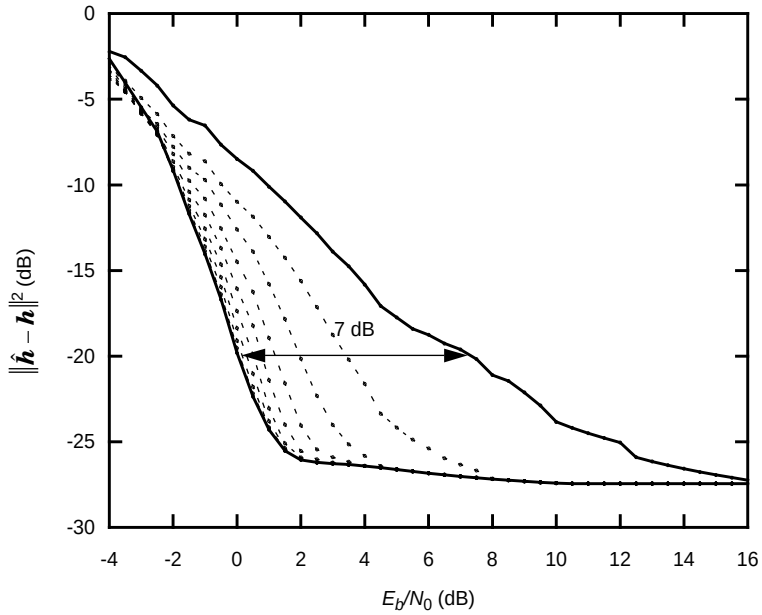

Fig. 42. Comparison between CMS and ECC-aware blind channel estimates.

consists of initializing the equalizer output to sign(r_k), *i.e.*, we make decisions on the transmitted codeword ignoring noise and ISI. In other words, at the first iteration the channel is estimated with sign(r_k) replacing $\tanh\left(\frac{L_k}{2}\right)$. The resulting WER for these initialization strategies is shown in Fig. 43. As we can see, even though the CMS algorithm can certainly help the convergence of the ECC-aware estimator at high SNR, it is not very helpful at low SNR. This happens because CMS operates in the uncoded domain, so in our example it "sees" an SNR of -4 dB. In other words, in the region where the ECC-aware estimator is most useful, the low SNR region, CMS is not able to produce reliable channel estimates.

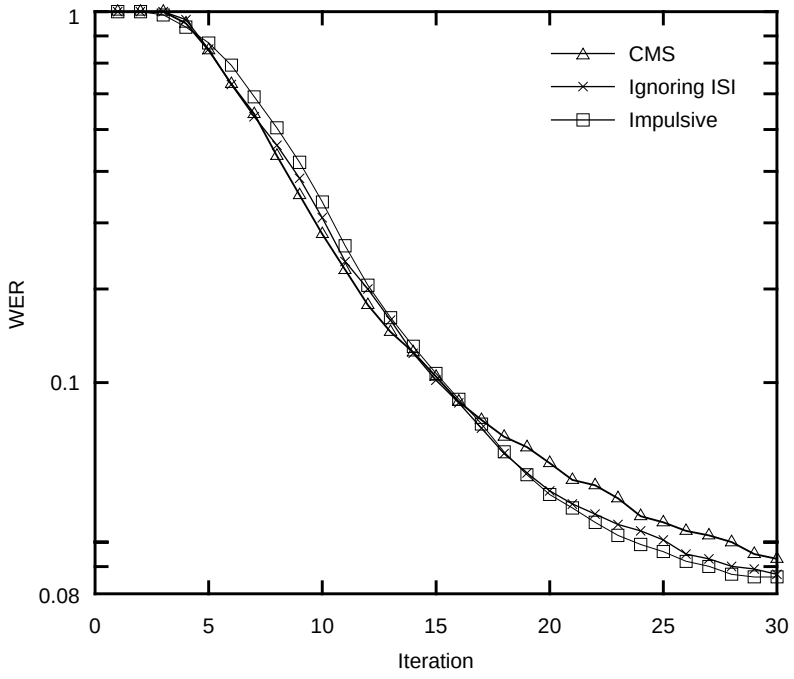

Fig. 43. WER for different initialization strategies.

7.6 Turbo Estimator

In this section, we provide simulation results that show that the SFE and the EW algorithm may be combined to form the turbo estimator (TE), a linear-complexity ECC-aware blind channel estimator. In fact, as seen before, using an equalizer for ECC-aware estimation is not significantly different than using an equalizer for turbo equalization. Since the SFE can be used for turbo equalization, it should be no surprise that the SFE may be used for ECC-aware estimation.

To assess the performance of the TE, we simulated the same scenario used in Fig. 28, *i.e.*, 100 blocks of $K = 2^{15}$ message bits were encoded with a rate 1/2 recursive systematic convolutional code with generator polynomial $P(D) = (1 + D^2) / (1 + D + D^2)$. The resulting codewords was interleaved and transmitted through the channel $\boldsymbol{h} = [0.227, 0.46, 0.688, 0.46, 0.227]$. The SFE used $M_1 = 9$, $M_2 = 5$, and the channel estimates were initialized with the impulsive initialization. The resulting channel estimation errors are shown in Fig. 44, while the BER performance of this system is shown in Fig. 45. It is interesting to see that the channel estimation error stops improving at 15 iterations, while the BER performance continues to improve until the 25-th iteration. Comparing Fig. 28 and Fig. 45, we see that the TE performs as well as the system with channel knowledge, although the TE converges more slowly.

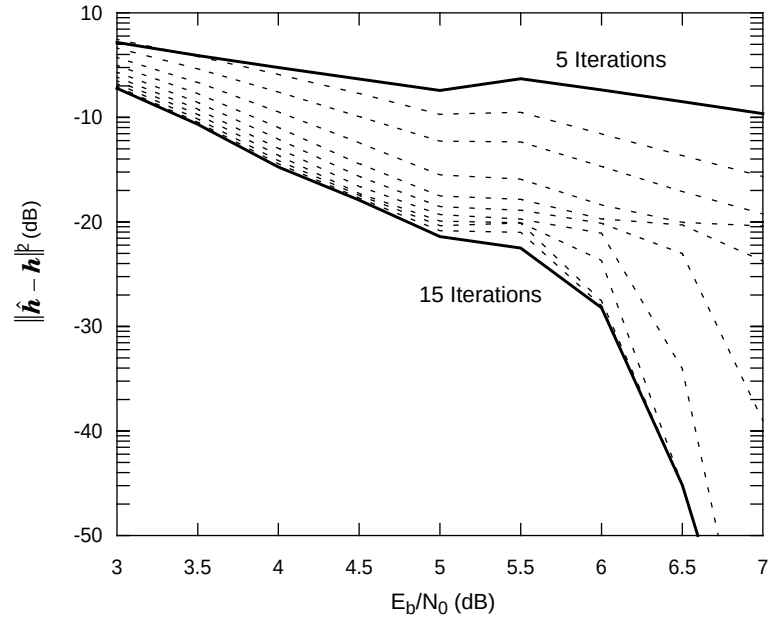

Fig. 44. Channel estimation errors for the SFE-based ECC-aware blind channel estimator.

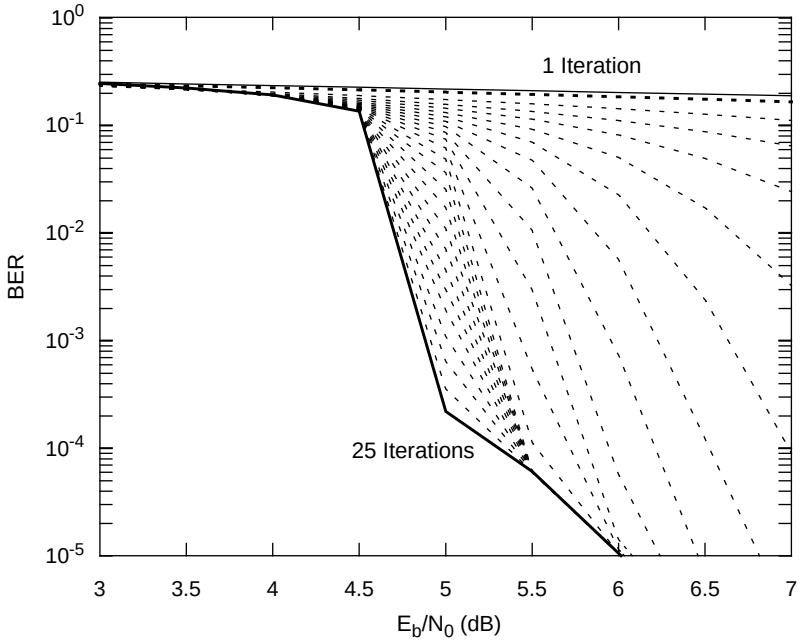

Fig. 45. BER performance of the SFE-based ECC-aware blind channel estimator.

7.7 Summary

In this chapter, we proposed an ECC-aware blind channel estimator, an iterative channel estimator that benefits from the presence of coding. We provided examples of the good quality of ECC-aware estimates. For instance, we showed that a turbo equalizer using ECC-aware estimates may perform almost as well as a turbo equalizer with channel knowledge. We also compared ECC-aware estimates to ECC-ignorant estimates, and showed that a gain of as much as 7 dB is possible for an estimation error of -20 dB. Furthermore, we showed that systems based on ECC-aware estimates may operate at very low SNR, where ECC-ignorant estimates yield poor performance. Finally, we proposed the turbo estimator, a linear-complexity ECC-aware channel estimator based on the EW algorithm of chapter 4 and the SFE of chapter 5. We showed that the SFE-based ECC-aware estimator retains all the desirable properties of the BCJR-based estimator.

CHAPTER 8

Conclusions

8.1 Summary of Contributions

In this work, we proposed and analyzed linear-complexity techniques for iterative channel estimation and equalization. Because of the way these techniques are designed, they may benefit from the presence of ECC, and hence may be used for turbo equalization and ECC-aware channel estimation.

In chapter 2, the problem of blind channel estimation for a coded system is introduced. A maximum-likelihood estimator is prohibitively complex, so this problem is normally divided in three subproblems: channel estimation, equalization and decoding. We discussed the divide and conquer approach, in which each of these subproblems is solved independently. We also discussed turbo equalizers and the EM algorithm. These are iterative algorithms that provide approximate solutions to otherwise intractable problems: respectively, joint decoding and equalization for known channels, and joint estimation and equalization for uncoded systems.

Turbo equalizers and the EM algorithm provide better performance than their non-iterative counterparts. Combining them into an iterative receiver that provides channel estimates that benefit from the presence of ECC is almost straightforward. However, these techniques suffer from complexity and convergence problems. The goal of this thesis was

to propose a system whose per-symbol computational complexity grows linearly with the number of coefficients in the system. The proposed system is less prone to misconvergence than systems based on the EM algorithm.

In chapter 3, we proposed the SEM algorithm, a linear complexity channel estimator [20] that performs almost as well as the EM algorithm, whose channel estimator has quadratic complexity. More importantly, the SEM is not intrinsically tied to an equalizer, so further complexity reduction is possible if the BCJR equalizer in the EM algorithm is replaced by a lower-complexity equalizer. We presented a detailed analysis of the SEM algorithm for a scalar channel, and compared the performance of the SEM and the EM algorithm for channels that introduce ISI. Simulations established that the performance is not significantly degraded with the SEM algorithm.

In chapter 4, we studied the convergence issues of the EM algorithm. We showed that in some cases of misconvergence of the EM algorithm, the resulting channel estimates may be seen as shifted versions of the actual channel. With this observation in mind, we developed the EW algorithm, a linear complexity channel estimator with better convergence properties than the EM algorithm [20].

In chapter 5, we addressed the complexity of the BCJR equalizer, which is used in the EM algorithm and in turbo equalizers. We discussed techniques that replace the BCJR equalizer by a linear equalizer and an ISI canceller. The output of the linear equalizer contains residual ISI, which is removed by the ISI canceller using the extrinsic information at the equalizer output. Most of the techniques proposed in the literature have

quadratic complexity. Some linear-complexity techniques have also been proposed. We proposed the SFE, a linear-complexity soft-output equalizer that is suited for iterative applications [52] and that outperforms existing linear-complexity equalizers.

In chapter 6, we studied the application of the SFE in turbo equalization. We discussed some issues arising in this application, and we showed that the SFE outperforms other linear-complexity equalizers in this context. We also discussed EXIT charts, and showed the charts of various equalizers.

Finally, in chapter 7, we proposed the turbo estimator (TE), an ECC-aware channel estimator that uses the fact that the transmitted sequence was encoded to improve blind and semi-blind channel estimates. These ECC-aware channel estimators may be seen as a combination of the EM or EW algorithm with a turbo equalizer. Therefore, using the SFE for equalization we obtain a linear-complexity ECC-aware estimator [17]. We showed that ECC-aware blind estimates may yield a BER performance similar to that of systems with channel knowledge. We also showed that ECC-aware blind estimates allow systems to operate at an SNR so low that other ECC-ignorant blind estimators fail. Because of these observations, ECC-aware blind estimators may be essential for blind systems to enjoy the full benefits of turbo equalization.

8.2 Directions for Future Research

Even though the EW algorithm improves the convergence of the iterative channel estimator, there is still a nonzero probability that the estimator will not converge to the correct channel estimates. A better understanding of the reasons for misconvergence is needed, and a globally convergent blind iterative channel estimator is yet to be determined.

The performance of iterative schemes is hard to determine, which has motivated the development of approximate analysis tools such as the EXIT charts. However, most of these tools are based on simulations. A purely theoretical tool would be of interest.

When comparing the SFE to a DFE, we have observed that sometimes hard feedback gives better performance than soft feedback, and sometimes the reverse happens. A deeper investigation of this behavior should be conducted. For instance, one could try to determine in which cases hard information works better than soft information, and in which cases the reverse happens.

The techniques proposed in this research were tested on general simulation channels. It would be interesting to test them in real world applications. One possibility that is currently under investigation in the Communication Theory Research Group at the Georgia Institute of Technology is the use of the SFE for equalization in magnetic recording channels. In magnetic recording systems, the received signal is normally filtered with a linear equalizer so that the cascade of the channel and the equalizer has a given impulse response. Normally, this impulse response is short enough that a BCJR equalizer may be used. However, as the recording density increases, so does the length of the impulse response of the cascade of the channel and the equalizer. Furthermore, the linear

equalizer introduces some noise enhancement and does not benefit from turbo equalization. The system under investigation would replace the linear equalizer and the BCJR equalizer by a single SFE. It is expected that the SFE-base system may even outperform the BCJR-based system, since the latter suffers from the noise enhancement introduced by the linear equalizer.

Finally, it is a well-known fact that blind channel estimators cannot account for delays. In other words, the sequence at the blind-equalizer output may be a delayed version of the transmitted sequence. If a delayed sequence is fed to the deinterleaver, the resulting sequence will be uncorrelated to the decoder output. In this work, this delay issue did not arise since we assumed the channel length to be known. In practical applications, however, this assumption is usually false. Therefore, a technique for resolving the delay must be found.

APPENDIX A

Computing Hard Scalar-Channel Estimates

Consider a scalar channel, where the received signal is written as $r_k = A\,a_k + n_k$. As seen in chapter 3, when hard information is used for estimating the gain and variance of this channel, we obtain the following asymptotic (as the number of observations tends to infinity) estimates:

$$\hat{A}_{i+1} = \mathrm{E}\left[r_k \operatorname{sign}\left(\frac{1}{2}\hat{L}_i r_k\right)\right], \tag{84}$$

$$\hat{\sigma}^2_{i+1} = \mathrm{E}\left[\left|r_k - \hat{A}_{i+1}\operatorname{sign}\left(\frac{1}{2}\hat{L}_i r_k\right)\right|^2\right]. \tag{85}$$

In this case, it is possible to find closed form formulas for $\hat{A}$ and $\hat{\sigma}^2$. In this appendix, we derive these formulas. For notational convenience, let $\lambda_k = A\,r_k / \sigma^2$.

The formulas are particularly simple for $\hat{\sigma}^2$, and may be expressed in closed form even if the gain is estimated using soft decisions. In fact, we may write

$$\begin{aligned}
\hat{\sigma}^2_{i+1} &= \mathrm{E}[\,|\,r_k - \hat{A}_{i+1}\,\operatorname{sign}(\lambda_k)\,|^2] \\
&= \mathrm{E}[\,|\,r_k\,|^2] - 2\hat{A}_{i+1}\,\mathrm{E}[r_k\,\operatorname{sign}(\lambda_k)] + \hat{A}^2_{i+1}\,\mathrm{E}[\operatorname{sign}(\lambda_k)^2] \\
&= A^2 + \sigma^2 - 2\hat{A}_{i+1}\,\hat{A}_{hard} + \hat{A}^2_{i+1},
\end{aligned} \tag{A-1}$$

where $\hat{A}_{hard}$ is the channel estimate computed with hard information, whose formula we derive in the sequel. For notational convenience, we drop the iteration index $i + 1$. Thus,

$$\begin{aligned}\hat{A}_{hard} &= \mathrm{E}[r_k \operatorname{sign}(\lambda_k)] \\ &= \mathrm{E}[\mathrm{E}[r_k \operatorname{sign}(\lambda_k) \mid a_k]] \\ &= \frac{1}{2}\mathrm{E}[\mathrm{E}[r_k \operatorname{sign}(\lambda_k) \mid a_k = 1]] + \frac{1}{2}\mathrm{E}[\mathrm{E}[r_k \operatorname{sign}(\lambda_k) \mid a_k = -1]]. \end{aligned} \quad \text{(A-2)}$$

Due to the symmetry of the distributions involved, as well as the symmetry of the decision function, both expected values in equation (A-2) are the same, so we may write

$$\begin{aligned}\hat{A} &= \mathrm{E}[\mathrm{E}[r_k \operatorname{sign}(\lambda_k) \mid a_k = 1]]. \\ &= A\mathrm{E}[\operatorname{sign}(\hat{L}\,A + \hat{L}\,n_k)] + \mathrm{E}[n_k \operatorname{sign}(\hat{L}\,A + \hat{L}\,n_k)] \\ &= A\mathrm{E}[\operatorname{sign}(A + n_k)] + \mathrm{E}[n_k \operatorname{sign}(A + n_k)], \end{aligned} \quad \text{(A-3)}$$

where the last equality follows from the fact that $\hat{L} > 0$, so that $\operatorname{sign}(\hat{L}\,x) = \operatorname{sign}(x)$.

The first term in (A-3) can be written as

$$\begin{aligned}\mathrm{E}[\operatorname{sign}(A + n_k)] &= 1\ \Pr[A + n_k > 0] - 1\ \Pr[A + n_k < 0] \\ &= 1 - 2\,\mathrm{Q}\left(\frac{A}{\sigma}\right). \end{aligned} \quad \text{(A-4)}$$

Likewise, the second term of equation (A-2) can be written as

$$\begin{aligned}\mathrm{E}[n_k \operatorname{sign}(A + n_k)] &= \frac{1}{\sqrt{2\pi}\sigma}\int_{-A}^{\infty} n_k \exp\left(-\frac{n_k^2}{2\sigma^2}\right) dn_k - \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{-A} n_k \exp\left(-\frac{n_k^2}{2\sigma^2}\right) dn_k \\ &= \sqrt{\frac{2}{\pi}}\sigma \exp\left(-\frac{A^2}{2\sigma^2}\right). \end{aligned} \quad \text{(A-5)}$$

Thus, we can write

$$\hat{A} = A - 2A\,Q\left(\frac{A}{\sigma}\right) + \sqrt{\frac{2}{\pi}}\sigma\exp\left(-\frac{A^2}{2\sigma^2}\right), \tag{A-6}$$

concluding our derivation.

Now it is well known that

$$Q(x) \le \frac{1}{\sqrt{2\pi}x}\exp(-x^2). \tag{A-7}$$

Thus, rewriting (A-6) as

$$\hat{A} = A - 2A\left[Q\left(\frac{A}{\sigma}\right) - \frac{1}{\sqrt{2\pi}A/\sigma}\exp\left(-\frac{A^2}{2\sigma^2}\right)\right], \tag{A-8}$$

and applying the bound in (A-8) with $x = A / \sigma$, we obtain that $\hat{A} \ge A$, so that the channel gain is always overestimated.

Also, note that if hard information is used for estimating the noise variance, we obtain $\hat{\sigma}^2 = A^2 + \sigma^2 - \hat{A}^2$ from (A-1). Now note that $\hat{\sigma}^2$ is the expected value of a positive number, and hence is itself a positive number. Thus, $A^2 + \sigma^2 - \hat{A}^2 \ge 0$. Combining the inequalities, we find that

$$\sqrt{A^2 + \sigma^2} \ge \hat{A} \ge A \tag{A-9}$$

if $\hat{A}$ is computed with hard information.

APPENDIX B

Computing the SFE Coefficients

To find the values of $\boldsymbol{f}$, $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ that minimize $\mathrm{E}[|z_k - a_k|^2]$, we rewrite (51) as

$$z_k = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}_k, \tag{B-1}$$

where $\boldsymbol{y}_k = [\boldsymbol{r}_k^{\mathrm{T}}, \tilde{\boldsymbol{a}}_k^{\mathrm{T}}, \bar{\boldsymbol{a}}_k^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{x} = [\boldsymbol{f}^{\mathrm{T}}, -\boldsymbol{g}_1^{\mathrm{T}}, -\boldsymbol{g}_2^{\mathrm{T}}]^{\mathrm{T}}$. Then, the MSE may be written as

$$\begin{aligned}\mathrm{E}[|z_k - a_k|^2] &= \mathrm{E}[|\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}_k - a_k|^2] \\ &= \boldsymbol{x}^{\mathrm{T}}\mathrm{E}[\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}]\boldsymbol{x} - 2\boldsymbol{x}^{\mathrm{T}}\mathrm{E}[\boldsymbol{y}_k a_k] + \mathrm{E}[|a_k|^2].\end{aligned} \tag{B-2}$$

From (B-2), it is easy to see that the MMSE solution satisfies

$$\mathrm{E}[\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}]\boldsymbol{x} = \mathrm{E}[\boldsymbol{y}_k a_k], \tag{B-3}$$

which can be rewritten, using the definition of $\boldsymbol{y}_k$ and $\boldsymbol{x}$, as

$$\begin{bmatrix} \mathrm{E}[\boldsymbol{r}_k\boldsymbol{r}_k^{\mathrm{T}}] & \mathrm{E}[\boldsymbol{r}_k\tilde{\boldsymbol{a}}_k^{\mathrm{T}}] & \mathrm{E}[\boldsymbol{r}_k\bar{\boldsymbol{a}}_k^{\mathrm{T}}] \\ \mathrm{E}[\tilde{\boldsymbol{a}}_k\boldsymbol{r}_k^{\mathrm{T}}] & \mathrm{E}[\tilde{\boldsymbol{a}}_k\tilde{\boldsymbol{a}}_k^{\mathrm{T}}] & \mathrm{E}[\tilde{\boldsymbol{a}}_k\bar{\boldsymbol{a}}_k^{\mathrm{T}}] \\ \mathrm{E}[\bar{\boldsymbol{a}}_k\boldsymbol{r}_k^{\mathrm{T}}] & \mathrm{E}[\bar{\boldsymbol{a}}_k\tilde{\boldsymbol{a}}_k^{\mathrm{T}}] & \mathrm{E}[\bar{\boldsymbol{a}}_k\bar{\boldsymbol{a}}_k^{\mathrm{T}}] \end{bmatrix} \begin{bmatrix} \boldsymbol{f} \\ -\boldsymbol{g}_1 \\ -\boldsymbol{g}_2 \end{bmatrix} = \begin{bmatrix} \mathrm{E}[\boldsymbol{r}_k a_k] \\ \mathrm{E}[\tilde{\boldsymbol{a}}_k a_k] \\ \mathrm{E}[\bar{\boldsymbol{a}}_k a_k] \end{bmatrix}. \tag{B-4}$$

Now, assume that $\mathrm{E}[\tilde{a}_k a_j] = \mathrm{E}[\bar{a}_k a_j] = \mathrm{E}[\tilde{a}_k \bar{a}_j] = 0$ when $k \neq j$. This seems reasonable, since $\tilde{a}_k$ and $\bar{a}_k$ are approximately equal to a_k and the transmitted symbols are uncorrelated. Furthermore, assume that $\mathrm{E}[|a_k|^2] = 1$. Using these assumptions and (61), which states that $\boldsymbol{r}_k = \mathbf{H}\boldsymbol{a}_k' + \boldsymbol{n}_k$, we find that the MMSE coefficients satisfy

$$\begin{bmatrix} \mathbf{H}\mathbf{H}^{\mathrm{T}} + \sigma^2\mathbf{I} & \alpha_1\mathbf{H}_1 & \alpha_2\mathbf{H}_2 \\ \alpha_1\mathbf{H}_1^{\mathrm{T}} & E_1\mathbf{I} & \mathbf{0} \\ \alpha_2\mathbf{H}_2^{\mathrm{T}} & \mathbf{0} & E_2\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{f} \\ -\boldsymbol{g}_1 \\ -\boldsymbol{g}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{B-5}$$

where $E_1 = \mathrm{E}[|\tilde{a}_k|^2]$, $E_2 = \mathrm{E}[|\bar{a}_k|^2]$, $\alpha_1 = \mathrm{E}[\tilde{a}_k a_k]$, and $\alpha_2 = \mathrm{E}[\bar{a}_k a_k]$. The vector $\boldsymbol{h}_0$ is the 0-th column of $\mathbf{H}$, where the columns of $\mathbf{H}$ are numbered as $\mathbf{H} = [\boldsymbol{h}_{-M_1}, \ldots, \boldsymbol{h}_{M_2+\mu}]$. Also, $\mathbf{H}_1 = [\boldsymbol{h}_{-M_1}, \ldots \boldsymbol{h}_{-1}]$ and $\mathbf{H}_2 = [\boldsymbol{h}_1, \ldots \boldsymbol{h}_{M_2+\mu}]$.

The last two block-rows in (B-5) yield

$$\boldsymbol{g}_1 = (\alpha_1 / E_1)\, \mathbf{H}_1^{\mathrm{T}} \boldsymbol{f} \tag{B-6}$$

$$\boldsymbol{g}_2 = (\alpha_2 / E_2)\, \mathbf{H}_2^{\mathrm{T}} \boldsymbol{f}. \tag{B-7}$$

Using these values, the first row of (B-5) may be written as

$$(\mathbf{H}\mathbf{H}^{\mathrm{T}} + \sigma^2\mathbf{I} - \frac{\alpha_1^2}{E_1}\mathbf{H}_1\mathbf{H}_1^{\mathrm{T}} - \frac{\alpha_2^2}{E_2}\mathbf{H}_2\mathbf{H}_2^{\mathrm{T}})\boldsymbol{f} = \boldsymbol{h}_0, \tag{B-8}$$

yielding

$$\boldsymbol{f} = (\mathbf{H}\mathbf{H}^{\mathrm{T}} - \frac{\alpha_1^2}{E_1}\mathbf{H}_1\mathbf{H}_1^{\mathrm{T}} - \frac{\alpha_2^2}{E_2}\mathbf{H}_2\mathbf{H}_2^{\mathrm{T}} + \sigma^2\mathbf{I})^{-1}\, \boldsymbol{h}_0, \tag{B-9}$$

which completes the derivation of the SFE coefficients.

Finally, assume that $z_k = Aa_k + v_k$, where A is a gain and v_k an equivalent noise with variance σ_v^2, assumed to be Gaussian and independent of a_k. In this case, $A = \mathrm{E}[z_k a_k]$. Using (B-1), this yields $A = \boldsymbol{x}^{\mathrm{T}}\mathrm{E}[\boldsymbol{y}_k a_k]$. However, as seen in (B-5), $\mathrm{E}[\boldsymbol{y}_k a_k] = [\boldsymbol{h}_0^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}}]^{\mathrm{T}}$. Thus, since $\boldsymbol{x} = [\boldsymbol{f}^{\mathrm{T}}, -\boldsymbol{g}_1^{\mathrm{T}}, -\boldsymbol{g}_2^{\mathrm{T}}]^{\mathrm{T}}$, we have that

$$A = \boldsymbol{f}^{\mathrm{T}}\boldsymbol{h}_0. \tag{B-10}$$

Furthermore, $\mathrm{E}[|z_k|^2] = A^2 + \sigma_v^2$. However, using (B-1), we may write $\mathrm{E}[|z_k|^2] = \boldsymbol{x}^{\mathrm{T}}\mathrm{E}[\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}]\boldsymbol{x}$. Then, using (B-3), we get that $\mathrm{E}[|z_k|^2] = \boldsymbol{x}^{\mathrm{T}}\mathrm{E}[\boldsymbol{y}_k a_k] = \boldsymbol{A}$. Thus,

$$\sigma_v^2 = A - A^2. \tag{B-11}$$

References

[1] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, Second Edition, Kluwer Academic Publishers, 1994.

[2] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, 1995.

[3] C. P. Williams and S. H. Clearwater, *Explorations in Quantum Computing*,Spinger-Verlag, New York, 1998.

[4] Poor, H. V., *An Introduction to Signal Detection and Estimation*, Second Edition, Springer-Verlag, 1994.

[5] L. Ljung and T. Söderström, *Theory and practice of recursive identification*, MIT Press, Cambridge, Mass., 1983.

[6] J. Ayadi, E. de Carvalho and D. T. M. Slock, "Blind and Semi-Blind Maximum Likelihood Methods for FIR Multichannel Identification," *IEEE ICASSP*, vol. 6, pp. 3185 -3188, 1998.

[7] S. Haykin, ed, *Blind Deconvolution*, Prentice-Hall, 1994.

[8] Z. Ding and Y. Li, *Blind equalization and identification*, Marcel Dekker, New York, 2001.

[9] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding," *IEEE International Conference on Commun.*, pp. 1064-1070, May 1993.

[10] D. J. MacKay, "Good Error Correcting Codes Based on Very Sparse Matrices," *IEEE T. Info. Theory*, p. 399-431, May 1999.

[11] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design and Iterative Decoding," *IEEE Trans. Information Theory*, vol. 44, no. 3, pp. 909-926, May 1998.

[12] J. Garcia-Frias and J. D. Villasenor, "Blind Turbo Decoding and Equalization," *IEEE Vehicular Technology Conference*, pp. 1881-1885, v. 3, May 1999.

[13] J. Garcia-Frias and J. D. Villasenor, "Combined Blind Equalization and Turbo Decoding," *IEEE Communications Theory Mini-Conference*, pp. 52-57, June 1999.

[14] K. -D. Kammeyer, V. Kühn and T. Peterman, "Blind and Nonblind Turbo Estimation for Fast Fading GSM Channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1718-1728, Sep. 2001.

[15] A. O. Berthet, B. S. Ünal and R. Visoz, "Iterative Decoding of Convolutionally Encoded Signals over Multipath Rayleigh Fading Channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1728-1743, Sep. 2001.

[16] P. Ha and B. Honary, "Improved Blind Detector," *IEEE Vehicular Technology Conference*, pp. 1196-1199, v. 2, May 2000.

[17] R. Lopes and J. Barry, "Exploiting Error-Control Coding in Blind Channel Estimation," *IEEE Global Communications Conference*, vol. 2, pp. 1317-1321, San Antonio, TX, November 2001.

[18] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier and A. Glavieux, "Iterative Correction of Intersymbol Interference: Turbo Equalization," *European Transaction on Telecommunications*, vol. 6, no. 5, pp 507-511, Sep.-Oct. 1995.

[19] D. Raphaeli and Y. Zarai, "Combined Turbo Equalization and Turbo Decoding," *IEEE Communications Letters*, vol. 2, no.4, pp.107-109, April 1998.

[20] R. R. Lopes and J. R. Barry, "Blind Iterative Channel Identification and Equalization," *IEEE International Conference on Communications*, pp. 2256-2260, Finland, June 2001.

[21] M. Feder and J. A. Catipovic, "Algorithms for Joint Channel Estimation and Data Recovery — Application to Equalization in Underwater Communications," *IEEE J. Oceanic Eng.*, vol. 16, no. 1, pp. 42–5, Jan. 1991.

[22] G. K. Kaleh and R. Vallet, "Joint Parameter Estimation and Symbol Detection for Linear or Nonlinear Unknown Channels", *IEEE Transactions on Communications*, vol. 42, no. 7, pp. 2406-2413, July 1994.

[23] C. Anton-Haro, J. A. R. Fonollosa and J. R. Fonollosa, "Blind Channel Estimation and Data Detection Using Hidden Markov Models," *IEEE Trans. Sig. Proc.*, vol. 45, no. 1, pp. 241-246, Jan 1997.

[24] V. Krishnamurthy, J. B. Moore, "On-Line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure," *EEE Trans. Sig. Proc.*, vol. 41, no. 8, pp. 2557-2573, Aug 1993.

[25] L. B. White, S. Perreau, P. Duhamel, "Reduced Complexity Blind Equalization for FIR Channel Input Markov Models", *IEEE International Conference on Communications,* vol. 2, pp.993-997, 1995.

[26] M. Shao and C. L. Nikias, "An ML/MMSE Estimation Approach to Blind Equalization", *ICASSP,* vol. 4, pp. 569-572, 1994.

[27] H. A. Cirpan and M. K. Tsatsanis, "Stochastic Maximum Likelihood Methods for Semi-Blind Channel Equalization," *Signal Processing Letters*, vol. 5, no. 1, pp. 1629-1632, Jan 1998.

[28] B. P. Paris, "Self-Adaptive Maximum-Likelihood Sequence Estimation," *IEEE Global Telecommunications Conf.*, vol. 4, pp. 92-96, 1993.

[29] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, pp. 284-287, March 1974.

[30] A. Glavieux, C. Laot and J. Labat, "Turbo Equalization Over a Frequency Selective Channel," *International Symp. on Turbo Codes and Related Topics*, Brest, France, pp. 96-102, Sep 1997.

[31] Z. Wu and J. Cioffi, "Turbo Decision Aided Equalization for Magnetic Recording Channels," *IEEE Global Communications Conference*, pp. 733-738, Dec. 1999.

[32] D. Raphaeli and A. Saguy, "Reduced Complexity APP for Turbo Equalization," *IEEE International Conf. Comm.*, vol. 3, pp. 1940-1943, 2002.

[33] A. Dejonghe and L. Vanderdorpe, "Turbo Equalization for Multilevel Modulation: an Efficient Low-Complexity Scheme," *IEEE International Conference on Communications*, vol. 3, pp. 1863-1867, 2002.

[34] J. Rößler, W. Gerstacker, A. Lampe, and J. Huber, "Matched-Filter- and MMSE-Based Iterative Equalization with Soft Feedback for QPSK Transmission," *International Zurich Seminar on Broadband Comm.*, pp. 19-1 - 19-6, February 2002.

[35] M. Tüchler, R. Koetter and A. C. Singer, "Turbo Equalization: Principles and New Results," *IEEE Transactions on Communications*, vol. 50, no. 5, pp. 754-767, May 2002.

[36] A. Berthet, R. Visoz and P. Tortelier, "Sub-Optimal Turbo-Detection for Coded 8-PSK Signals over ISI Channels with Application to EDGE Advanced Mobile Systems," *IEEE International Symp. on Personal, Indoor and Mobile Radio Comm.*, pp. 151-157, Sep. 2000.

[37] R. Visoz, A. O. Berthett and J. J. Boutros, "Reduced-Complexity Iterative Decoding and Channel Estimation for Space Time BICM over Frequency-Selective Wireless Channels," *IEEE Symposion on Personal, Indoor and Mobile Radio Communications*, vol. 3, pp. 1017 -1022, 2002.

[38] C. Fragouli, N. Al-Dhahir and W. Turin, "Prefiltered M-BCJR Equalization of Space-Time-Coded Transmission over Frequency-Selective Channels," *Conf. Info. Sciences and Systems*, pp. 841-845, March 2001.

[39] A. A. Chang and G. Wornell, "A Class of Block-Iterative Equalizers for Intersymbol Interference Cancellation: Fixed Channels Results," *IEEE Trans. on Comm.*, vol. 49, no. 11, pp. 1966-1976, Nov. 2001.

[40] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm", *Journal of the Royal Statistics Society*, vol. 39, no. 1, pp 1-38, 1997.

[41] L. E. Baum *et al.* "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains", *Annals of Mathematics Statistics*, vol. 41, pp. 164-171, 1970.

[42] W. Ryan, "A Turbo Code Tutorial," unpublished, http://www.ece.arizona.edu/~ryan/turbo2c.pdf

[43] G. Stüber, *Principles of Mobile Communication*, Kluwer Academic Publishers, 2nd Edition, 2001.

[44] J. A. C. Bingham, "Multicarrier Modulation for Data Transmission: An Idea whose Time has Come," *IEEE Communications Magazine*, May 1990, pp. 5-14.

[45] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. on Comm.*, vol. 40, pp. 1727-1737, Oct. 2001.

[46] R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, 1968.

[47] H.-A. Loeliger and D. Arnold, "On the Information Rate of Binary-Input Channels with Memory," *IEEE Int. Symp. Inform. Theory*, p. 164, July 2002.

[48] J. Proakis, *Digital Communications*, McGraw-Hill, 1995.

[49] http://spib.rice.edu/spib/data/signals/comm/microwave/chan14a

[50] K. R. Narayanan, X. Wang and G. Yue, "LDPC code design for turbo equalization," *IEEE Information Theory Workshop*, pp 57 -60, 2002.

[51] Z. Ding and J. Liang, "A Cumulant Matrix Subspace Algorithm for Blind Single FIR Identification," *IEEE Transactions on Signal Processing*, vol. 49, no. 2, pp. 325-333, Feb 2001.

[52] R. Lopes and J. Barry, "Soft-Output Decision-Feedback Equalizer with Priors," *IEEE Global Communications Conference*, 2003.

VITA

Renato Lopes is from Campinas, Brazil. He received Bachelor's and Master's degrees from the State University of Campinas (UNICAMP) in 1995 and 1997, respectively, both in Electrical Engineering. In 1993 he worked with the Telephone Company of Tampere, Finland, in the area of GSM services. In 1994 he worked with PROMON Eletrônica, Campinas, Brazil, in the department responsible for testing switchboards. He received the Ph.D. degree from the Georgia Institute of Technology in 2003. During his Ph.D., he was supported by CAPES, and was also a teaching and research assistant at the Georgia Institute of Technology. He will join UNICAMP as a post-doctorate fellow. His research interest are in the general area of communications theory, with emphasis in equalization, channel estimation and iterative techniques for communications.