

Turbo Equalization

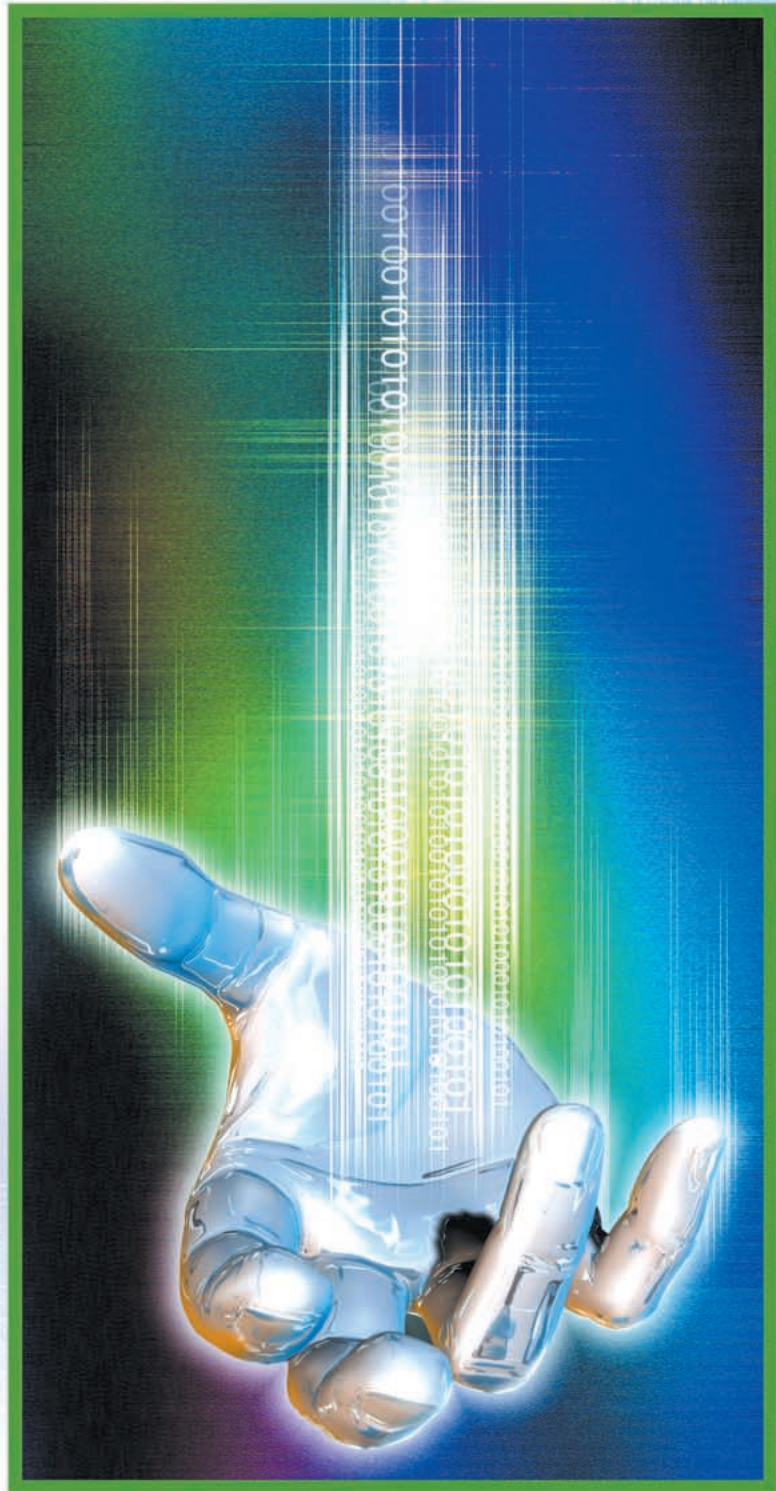
An iterative equalization and decoding technique for coded data transmission.

*Ralf Koetter,
Andrew C. Singer,
and Michael Tüchler*

Capitalizing on the tremendous performance gains of turbo codes and the turbo decoding algorithm, turbo equalization is an iterative equalization and decoding technique that can achieve equally impressive performance gains for communication systems that send digital data over channels that require equalization, i.e., those that suffer from intersymbol interference (ISI). In this article, we discuss the turbo equalization approach to coded data transmission over ISI channels, with an emphasis on the basic ideas and some of the practical details. The original system introduced by Douillard et al. can be viewed as an extension of the turbo decoding algorithm by considering the effect of the ISI channel as another form of error protection, i.e., as a rate-1 convolutional code.

Background and Organization

Graphical models for turbo codes (and low-density, parity-check (LDPC) codes), together with the various iterative algorithms for



© IMAGESTATE

decoding them, have provided substantial insights into the dramatic performance improvements achievable through their use [1]–[4]. For an overview about graphical models, we also refer to [5]. The flurry of research in related topics over the last decade has produced a number of communications and signal-processing algorithms that leverage turbo decoding approaches to provide similar gains in performance for a wide array of problems [6]–[14]. The original system introduced in [9] leveraged the ideas of the turbo decoding algorithm to the related problem of equalization and decoding. We seek to provide an overview of the turbo equalization approach, with an algorithmic description and intuitive explanation of each of the steps involved in designing such a communication system.

In the following section, we provide a brief overview of the turbo equalization approach. Then, we present a basic system model for a coded data transmission system together with a notional system example. Next, optimal detection strategies are discussed, followed by separate equalization and decoding methods. The turbo equalization algorithm is described in more detail, together with some measures of system performance for our example. Extensions and concluding remarks are presented in the last section.

Overview

In this section, we provide a high-level overview of turbo equalization, placing an emphasis on the concepts involved and delay a more mathematical development to subsequent sections of the article. The focus of our discussion will be the communication link depicted in Figure 1, which contains a system configuration for a digital transmitter as part of a communication link. These basic elements are contained in most practical communication systems and are essential components of a transmitter such that turbo equalization can be used in the receiver.

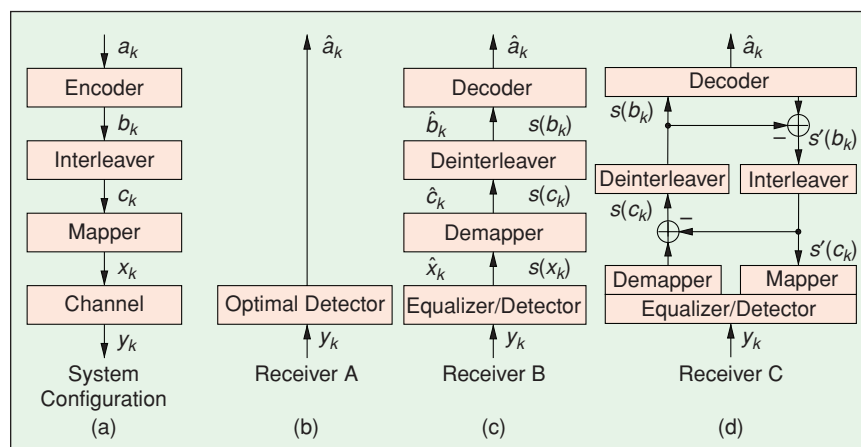
The role of the encoder, which is the first block in Figure 1(a), is to take the binary data sequence to be transmitted as input and produce an output that contains

not only this data but also additional redundant information that can be used to protect the data of interest in the event of errors during transmission. There are a wide variety of practical methods for introducing such redundancy in the form of an error control code (ECC) (also referred to as forward error correction); however, we will assume that a convolutional code is used for our purposes. The goal of forward error correction is to protect the data from the possibility of random single-bit errors or short bursts of errors that might occur in the data stream as a result of additive noise in the transmission or receiver errors. To ensure that such errors appear random and to avoid long error bursts, an interleaver is used to randomize the order of the code bits prior to transmission. This process is completely reversible and is simply mirrored in the receiver. Finally, the permuted code bits are then converted into electrical signal levels that can be modulated either at baseband or onto a carrier for transmission over a passband channel. Such modulation could take a variety of forms in such diverse applications as wired or wireless transmission, optical communications, optical data storage, magnetic recording, or even acoustic communication systems. The process of mapping binary code bits into channel symbols suitable for modulation is depicted by the mapper in Figure 1.

The traditional methods of data protection used in ECC do not work well when the channel over which the data is sent introduces additional distortions in the form of ISI. When the channel is bandlimited or for other reasons is dispersive in nature, then the receiver will, in general, need to compensate for the channel effects prior to employing a standard decoding algorithm for the ECC. Such methods for channel compensation are typically referred to as channel equalization. Even when the actual transmission medium is nondispersive, often the transmit-and-receive filtering that takes place in a practical system gives rise to sufficient ISI such that equalization becomes necessary.

Given observations of the received data, the receiver now has essentially one task to complete: estimate the

data that was transmitted. To do this optimally, in terms of minimizing the bit error rate (BER), the receiver must find the set of transmitted bits that are most probable, given knowledge of the complex statistical relationship between the observations and the transmitted bits. Such a receiver, as depicted in Figure 1(b) as receiver A, takes into account the ECC, the interleaver, the symbol mapping, and knowledge of the channel. With so many factors involved, the resulting statistical relationship rapidly becomes difficult to manage in an efficient manner. As such, in most practical systems, receiver A is simply infeasible.



▲ 1. System configuration and three receiver structures: the optimal detector (receiver A), one-time equalization and decoding using hard or soft decisions (receiver B), and turbo equalization (receiver C).

ble, as it amounts to essentially trying to fit all possible sequences of transmitted bits to the received data, a task whose complexity grows exponentially in the length of the data transmitted.

The way that most practical receivers have been designed is to first process the received observations to account for the effects of the channel and to make estimates of the transmitted channel symbols that best fit the observed data. A number of criteria for performance have been used for such equalizers, ranging from those attempting to simply invert the channel (so-called zero forcing equalizers) to linear and nonlinear equalizers based on minimizing a mean-squared error (MSE) metric to even those that are symbol-error-rate (SER) optimal by maximizing the likelihood of the observations given the channel and data model. These equalization methods constitute the first step in receiver B from Figure 1(c). Once the transmitted channel symbols have been estimated, they can be demapped into their associated code bits, deinterleaved, and then decoded using a BER optimal decoder for the ECC. The most straightforward way to implement this separate equalization and decoding process is for the equalizer to make hard decisions as to which sequence of channel symbols were transmitted and for these hard decisions to be mapped into their constituent binary code bits. These binary code bits can then be processed with the decoder for the ECC. The process of making hard decisions on the channel symbols actually destroys information pertaining to how likely each of the possible channel symbols might have been, however. This additional “soft” information can be converted into probabilities that each of the received code bits takes on the value of zero or one that, after deinterleaving, is precisely the form of information that can be exploited by a BER optimal decoding algorithm. Many practical systems use this form of soft-input error control decoding by passing soft information between an equalizer and decoding algorithm.

The remarkable performance of turbo codes makes it clear that the soft information need not only flow in one direction. Once the error control decoding algorithm processes the soft information it can, in turn, generate its own soft information indicating the relative likelihood of each of the transmitted bits. This soft information from the decoder could then be properly interleaved and taken into account in the equalization process, creating a feedback loop between the equalizer and decoder, through which each of the constituent algorithms communicates its beliefs about the relative likelihood that each given bit takes on a particular value. This process is often termed “belief propagation” or “message passing” and has a number of important connections to methods in artificial intelligence, statistical inference, and graphical learning theory. The feedback loop structure described here and depicted in receiver C in Figure 1(d) is essentially the process of turbo equalization.

The flurry of research in related topics over the last decade has produced a number of communications and SP algorithms that leverage turbo decoding approaches to provide similar gains in performance for a wide array of problems.

While the process of equalization and decoding through the feedback loop structure of receiver C is essentially complete, it is important to consider the effect that the soft information generated from one bit in one of the constituent algorithms (equalizer or decoder) will have on other bits in the other constituent algorithm. When processing soft information as an input to the equalizer or decoder, it is assumed that the soft information about each bit (or channel symbol) is an independent piece of information. This enables simple, fast algorithms to be used for each of the equalizer and decoder. If the decoder formulates its soft information about a given bit, based on soft information provided to it from the equalizer about exactly the same bit, then the equalizer cannot consider this information to be independent of its channel observations. In effect, this would create a feedback loop in the overall process of length two: the equalizer informs the decoder about a given bit, and then the decoder simply reinforces the equalizer what it already knows. To avoid such short cycles in the feedback and in hopes of avoiding local minima and limit cycle behavior in the process, when soft information is passed between constituent algorithms, such information is never formed based on the information passed into the algorithm concerning the same bit. Basically, this amounts to the equalizer only telling the decoder new information about a given bit based on information it gathered from distant parts of the received signal (thanks to the interleaver). Similarly, the decoder only tells the equalizer information it gathered from distant parts of the encoded bit stream. As a result, the iterative equalization and decoding process can continue for many iterations before cycles are introduced, which eventually limits further improvements. This process of only passing “extrinsic information” between constituent decoders is essential to the performance of turbo decoding algorithms.

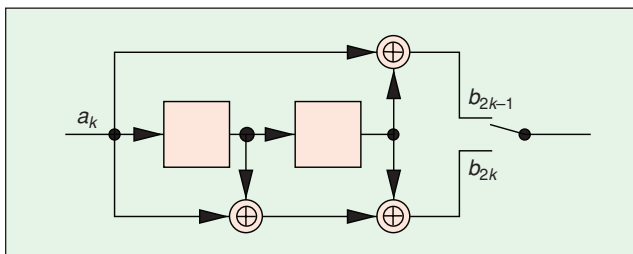
System Model

We begin with the goal of any communication system, which is to reliably transmit data over a given channel. As depicted in the transmission model in Figure 1(a), the

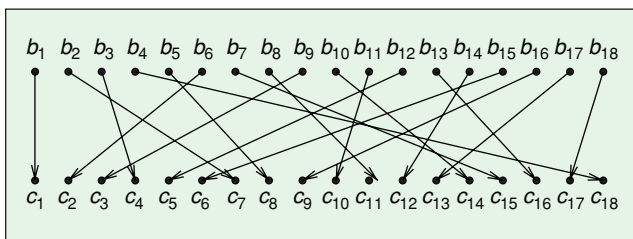
job of the transmitter is to send a stream of K binary data $a_k, k = 1, 2, \dots, K$, over the channel in a manner that will enable a receiver to correctly recover the original data stream with high probability. This is done in several steps. First, the binary data is protected from potential errors in the transmission process by introducing redundancy in the form of an ECC, producing a longer sequence of N (binary) code bits $b_k, k = 1, 2, \dots, N$. We are interested in the case of binary codes, i.e., the alphabet of the a_k and the b_k is $\{0, 1\}$ and addition (using the sign \oplus) and multiplication are performed modulo-2.

The rate $R = K/N$ of the code, which lies between zero and one, specifies the amount of added redundant information. In this article, we use a rate-1/2 convolutional code given by the generator $(1+D^2 \ 1+D+D^2)$ [15]. With this rate, K data bits a_k are encoded to $2K$ code bits b_k . Thus, the redundant information are K extra code bits. An encoder circuit for this code is depicted in Figure 2. We assume that the two delay elements in this circuit are zero at the beginning of the encoding process (time $k=0$) and at the end (time $k=K$). Obviously, the last two data bits a_{K-1} and a_K must be fixed to zero to achieve the latter assumption, which implies a small rate loss. This loss can be controlled by considering long sequences and can be avoided with tail-biting encoding [16].

To improve the performance of the ECC by spreading out any burst errors that might occur in the channel, an interleaver is used to scramble the data sequence b_k , creating the binary sequence c_k . The interleavers used in this article are the so-called S -random interleavers [17]. They randomly spread the bits b_k with the only restriction that each pair in a group of S consecutive bits (b_k, \dots, b_{k+S-1}) must be at least S indices apart after interleaving. Figure 3 depicts a three-random interleaver for 18 code bits.



▲ 2. Encoder of a convolutional code, where each incoming data bit a_k yields two code bits (b_{2k-1}, b_{2k}) via $b_{2k-1} = a_k \oplus a_{k-2}$ and $b_{2k} = a_k \oplus a_{k-1} \oplus a_{k-2}$.



▲ 3. A 3-random interleaver for 18 code bits.

The code bits c_k need to be physically modulated over the communication channel, which is typically done by grouping together code bits c_k into short blocks of, say, q bits each and mapping them onto a modulation waveform, or channel symbol, x_k . The modulation could be in the form of binary phase shift keying (BPSK), where a given pulse shape is modulated with either a $+1$ or -1 , i.e., the bit $c_k \in \{0, 1\}$ is mapped to a symbol x_k as $x_k = (-1)^{c_k}$. In this case, $q=1$. An example with $q=2$ is quadrature phase shift keying (QPSK) in which $x_k = (-1)^{c_{2k-1}} + j(-1)^{c_{2k}}$. Such complex modulation symbols naturally arise in the context of passband communication systems, in which magnitude and phase can be used to convey complex values easily. For simplicity, we will focus here on BPSK and refer to [13] for higher-order modulations.

The series of transmit pulse shapes modulated with the symbols $x_k, k = 1, 2, \dots, N$, is transmitted over a linear and time-invariant (or slowly time varying) communication channel with known channel impulse response (CIR). A coherent symbol-spaced receiver front end with precise knowledge of the signal phase and symbol timing receives the transmitted waveforms, which are distorted by the channel and by white Gaussian noise added in the receiver front end. The received waveforms are passed through the receive filter, which is matched to the transmit pulse shape and the CIR. Sampling the receive filter output yields a sequence of samples y_k given by

$$y_k = n_k + \sum_{l=0}^L h_l x_{k-l}, \quad k = 1, 2, \dots, N, \quad (1)$$

where the real-valued coefficients h_l are the sampled impulse response of the cascade of the transmit filter, the channel, and the receive filter. This response is assumed to be of finite length $L+1$, i.e. $h_l = 0$, for $l > L$. The symbols $x_k, k < 1$, transmitted prior to x_1 , are assumed to be zero.

Using BPSK modulation yields that the x_k and the channel output $\sum_{l=0}^L h_l x_{k-l}$ are real valued such that it suffices to consider only the real part of the noise added in the receiver front end. Thus, provided that the receive filter satisfies the Nyquist criterion, the noise samples n_k are independent and identically distributed (IID) real-valued Gaussian noise samples distributed with $p(n) = \exp(-n^2/(2\sigma^2))/\sqrt{2\pi\sigma^2}$.

A tapped delay line circuit implementing the equivalent time-discrete model (1) for transmission of data in a bandlimited, additive noise channel is depicted in Figure 4 for $L=2$. The example channel used in this article is a length-three channel taken from [18], with the coefficients $h_0 = 0.407, h_1 = 0.815, h_2 = 0.407$. We will assume that these coefficients are known to the receiver and do not vary in time. For this channel, the channel law (1) can be expressed in matrix form $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}$, where the length- N vectors \mathbf{y} and \mathbf{n} and the $N \times N$ matrix \mathbf{H} are given by

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & 0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_2 & h_1 & h_0 & 0 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & \cdots & 0 & h_2 & h_1 & h_0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix}.$$

The symbols x_k from the alphabet $\{+1, -1\}$ have unit average power which, at the receiver input, yields the signal-to-noise ratio (SNR)

$$\text{SNR} = \frac{\text{signal power}}{\text{noise power}} = \frac{\sum_{l=0}^L |h_l|^2}{\sigma^2} = \frac{1}{\sigma^2}.$$

This transmission model gives rise to serious distortions due to ISI. Nevertheless, the information theoretic limits of transmission through this channel are well understood [19]–[22]. For example, it is possible to numerically compute the largest rate for which we can transmit reliably over this channel given that the symbols x_k are binary as well as independent and identically distributed (IID) [21]. Given the BPSK alphabet $\{+1, -1\}$, this rate is at most 1 bit per channel use. This bound is attained when the SNR approaches ∞ . In our example, where $R=1/2$ bits are transmitted per channel use, an SNR of at least -1.6 dB is required for reliable data transmission. The question to be answered in the remainder of this article is therefore: How can we design a receiver with tractable computational complexity, for a fixed finite sequence length N , that can achieve a reasonably small error rate at an SNR as close as possible to -1.6 dB?

It is the job of the receiver to estimate the data that was transmitted, making use of knowledge of how the channel has corrupted the data together with the available redundancy that has been introduced to protect the data, in the form of the ECC. While the ECC alone would protect the data from additive noise, when the channel introduces ISI adjacent channel symbols are smeared together, introducing additional dependencies among the transmitted channel symbols. Generally, the problem of mitigating the effects of an ISI channel on the transmitted data is called “equalization” or “detection,” while the subsequent problem of recovering the data bits from the equalized symbol stream, making use of the ECC, is called “decoding.”

For complexity reasons, these problems have traditionally been considered separately, with limited interaction between the two blocks. As such, substantial performance degradation is typically induced through the separation of these inherently dependent tasks. The main contribution of much of the work in turbo equalization to date has been to enable feasible approaches to jointly solving the equalization and decoding tasks.

As a result, the performance gap between an optimal joint decoding and equalization and that achievable through systems with practical complexity has been narrowed in a manner similar to that of near Shannon-limit communications using turbo codes [23].

Optimal Detection

A natural starting point for the developments in this article is the optimal receiver achieving the minimum probability of error $P(a_k \neq \hat{a}_k)$ for each data bit a_k . It is well known that this is achieved by setting \hat{a}_k to that value $a \in \{0, 1\}$, which maximizes the a posteriori probability (APP) $P(a_k = a | \mathbf{y})$ given the observed sequence \mathbf{y} , i.e.,

$$\hat{a}_k = \underset{a \in \{0,1\}}{\operatorname{argmax}} P(a_k = a | \mathbf{y}). \quad (2)$$

An algorithm that achieves this task is commonly referred to as a maximum a-posteriori probability (MAP) algorithm, corresponding to “receiver A” in Figure 1(b).

Where binary quantities are concerned, such as the bits a_k , it is convenient to work with log-likelihood ratios (LLRs) rather than probabilities. The LLR for a binary variable a is defined as

$$L(a) = \ln \frac{P(a=0)}{P(a=1)}. \quad (3)$$

The LLR $L(a)$ contains the same information as any of the two probabilities for $P(a=0)$ or $P(a=1)$. In particular, we find that the sign of $L(a)$ determines whether $P(a=0)$ is larger or smaller than $P(a=1)$ with the special case $P(a=0) = P(a=1) = 1/2$ when $L(a)=0$. Similarly, we can define a conditional LLR of a binary random variable a given y :

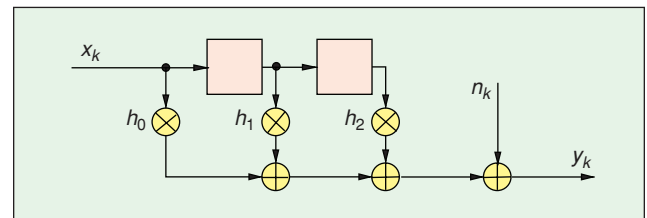
$$L(a|y) = \ln \frac{P(a=0|y)}{P(a=1|y)}. \quad (4)$$

Thus, the decision rule (2) can be written as

$$\hat{a}_k = \begin{cases} 0, & L(a_k | \mathbf{y}) \geq 0 \\ 1, & L(a_k | \mathbf{y}) < 0. \end{cases} \quad (5)$$

The main problem with the MAP approach is that computing the APPs is computationally demanding, since \mathbf{y} depends on the entire sequence $\mathbf{a} = (a_1 a_2 \dots a_K)^T$ of data bits a_k as follows:

$$P(a_k = a | \mathbf{y}) = \sum_{\forall \mathbf{a}: a_k = a} P(\mathbf{a} | \mathbf{y}) = \sum_{\forall \mathbf{a}: a_k = a} \frac{p(\mathbf{y} | \mathbf{a}) P(\mathbf{a})}{p(\mathbf{y})}. \quad (6)$$



▲ 4. Tapped delay line circuit of the channel model (1) for $L = 2$.

The probability $P(\mathbf{a})$ is the a-priori probability of the sequence \mathbf{a} , which can be used in (6) to include knowledge about the source producing the bits a_k . Usually, the bits a_k are assumed independent, i.e., the joint probability $P(\mathbf{a})$ factors as $\prod_{n=1}^K P(a_k)$. Applying the APP decomposition in (6) and that of $P(\mathbf{a})$ to the conditional LLR $L(a_k|\mathbf{y})$ yields the following:

$$\begin{aligned} L(a_k|\mathbf{y}) &= \ln \frac{\sum_{\mathbf{a}:a_k=0} p(\mathbf{y}|\mathbf{a}) \prod_{i=1}^K P(a_i)}{\sum_{\mathbf{a}:a_k=1} p(\mathbf{y}|\mathbf{a}) \prod_{i=1}^K P(a_i)} \\ &= \ln \frac{\sum_{\mathbf{a}:a_k=0} p(\mathbf{y}|\mathbf{a}) \prod_{i=1:i \neq k}^K P(a_i)}{\sum_{\mathbf{a}:a_k=1} p(\mathbf{y}|\mathbf{a}) \prod_{i=1:i \neq k}^K P(a_i)} + L(a_k) \\ &= L_{\text{ext}}(a_k|\mathbf{y}) + L(a_k). \end{aligned} \quad (7)$$

The quantity $L_{\text{ext}}(a_k|\mathbf{y})$ is the extrinsic information about a_k contained in \mathbf{y} . It adds to the a priori information $L(a_k)$ about a_k . Extrinsic LLRs will play a crucial role in the turbo equalization setup.

Since the bits a_k are usually assumed to be uniformly distributed, i.e., they take on the values 0 or 1 equally likely, we have that $L(a_k)=0$.

As seen in (6), receiver A is impractical for large block lengths K because of the exponentially growing number 2^K of terms $p(\mathbf{y}|\mathbf{a})$. In certain special cases, i.e., if the memory in the channel and in the interleaved coded sequence is only moderate, it is possible to organize the computation in a significantly more efficient fashion. Nevertheless, the number of operations per information bit grows exponentially in the sum of channel and code memory, which in many cases renders the problem intractable. In this situation, turbo equalization offers suboptimal alternatives to the optimal receiver, achieving comparable performance while requiring significantly reduced complexity. We will proceed by first developing the basic components of a turbo equalizer in a setting that separates the equalization and decoding tasks.

Separate Equalization and Decoding

Since a MAP algorithm for optimal joint equalization and decoding is usually not feasible, a standard approach to reducing the computational burden of the receiver is to split the detection problem into the two subproblems: equalization and decoding. This strategy is illustrated in “receiver B” depicted in Figure 1. One implementation communicates estimates \hat{x}_k , \hat{c}_k , and \hat{b}_k from the same alphabet ($\{0, 1\}$ or $\{-1, +1\}$) as x_k , c_k , and b_k , respectively, from the equalizer to the decoder. An alternative is to communicate reliability or soft information $s(x_k)$, $s(c_k)$, and $s(b_k)$, which provides more information on the relative likelihood that x_k , c_k , or b_k take on any particular value.

We will consider two distinct families of algorithms for the subproblem of equalization, namely, trellis-

based methods and equalization methods based on linear filtering. Typical trellis-based approaches include MAP symbol detection [24], [25] and maximum-likelihood (ML) sequence detection [18], [26]. A MAP symbol detector simply applies rule (2) while ignoring the effect of the code; i.e., it sets the symbol estimate \hat{x}_k to that symbol x from $\{+1, -1\}$, which maximizes the APP $P(x_k=x|\mathbf{y})$.

An ML sequence detector computes an estimate $\hat{\mathbf{x}}$ of the entire sequence \mathbf{x} , which maximizes the likelihood $p(\mathbf{y}|\mathbf{x})$. Both problems seem intractably complex because of the huge number of terms $p(\mathbf{y}|\mathbf{x})$ to be summed up as in (6) or to be maximized over for ML sequence detection. However, we will describe here a trellis-based approach in particular for MAP symbol detection to show that these problems can be solved efficiently.

MAP Symbol Detection

Consider the tapped delay line model of the transmitter, channel, and receive filter depicted in Figure 4. Assuming an impulse response length of $L+1$, the tapped delay line contains L delay elements. Thus, given a binary input alphabet $\{+1, -1\}$ the channel can be in one of 2^L states r_i , $i = 1, 2, \dots, 2^L$, corresponding to the 2^L different possible contents of the delay elements. We denote by $\mathcal{S} = \{r_1, r_2, \dots, r_{2^L}\}$ the set of possible states. At each time instance $k = 1, 2, \dots, N$ the state of the channel is a random variable $s_k \in \mathcal{S}$. It is an important property of the memory present in the system that, given s_k , the state s_{k+1} can only assume one of two values corresponding to a $+1$ or -1 being fed into the tapped delay line at time k . The possible evolution of states can thus be elegantly described in form of a trellis diagram. Any path through the trellis corresponds to a sequence of input and output symbols read from the branch labels, where the output symbol v_k at time k is the noise-free output of the channel model (1):

$$v_k = \sum_{l=0}^L h_l x_{k-l}.$$

The trellis for the channel of Figure 4 is depicted in Figure 5. A branch of the trellis is a four-tuple $(i, j, x_{i,j}, v_{i,j})$ such that state $s_{k+1} = r_j$ at time $k+1$ can be reached from state $s_k = r_i$ at time k with input $x_k = x_{i,j}$ and output $v_k = v_{i,j}$, where $x_{i,j}$ and $v_{i,j}$ are uniquely identified by the index pair (i, j) . The set of all index pairs (i, j) corresponding to valid branches is denoted \mathcal{B} . For the trellis in Figure 5, the set \mathcal{B} is as follows:

$$\mathcal{B} = \{(00), (01), (12), (13), (20), (21), (33), (32)\}.$$

The trellis description turns out to be extremely useful in computing the APPs $P(x_k|\mathbf{y})$. Motivated by our approach of separating the equalization from the decod-

ing task, we assume that the random variables x_k are IID, i.e., $P(\mathbf{x})$ factors into $\prod_{k=1}^N P(x_k)$ and x_k takes on values $+1$ and -1 equally for all k . To derive an efficient algorithm to compute $P(x_k|\mathbf{y})$, we start by computing the probability that the transmitted sequence path in the trellis contained the branch $(i, j, x_{i,j}, v_{i,j})$ at time k , i.e., $P(s_k = r_i, s_{k+1} = r_j|\mathbf{y})$. This APP can be computed efficiently with the forward/backward algorithm [24], [27], which is based on a suitable decomposition of the joint distribution $p(s_k, s_{k+1}, \mathbf{y})$ given by

$$p(s_k, s_{k+1}, \mathbf{y}) = p(\mathbf{y}) \cdot P(s_k, s_{k+1}|\mathbf{y}). \quad (8)$$

The sequence \mathbf{y} in $p(s_k, s_{k+1}, \mathbf{y})$ can be written as $p(s_k, s_{k+1}, (y_1, \dots, y_{k-1}), y_k, (y_{k+1}, \dots, y_N))$. Applying the chain rule for joint probabilities, i.e., $P(a, b) = P(a)P(b|a)$, to this expression yields the following decomposition of $p(s_k, s_{k+1}, \mathbf{y})$:

$$\underbrace{p(s_k, y_1, \dots, y_{k-1})}_{\alpha_k(s_k)} \underbrace{p(s_{k+1}, y_k|s_k)}_{\gamma_k(s_k, s_{k+1})} \underbrace{p(y_{k+1}, \dots, y_N|s_{k+1})}_{\beta_{k+1}(s_{k+1})}.$$

The term $\alpha_k(s)$ can be computed via the recursion:

$$\alpha_k(s) = \sum_{s' \in \mathcal{S}} \alpha_{k-1}(s') \gamma_{k-1}(s', s) \quad (9)$$

with the initial value $\alpha_0(s) = P(s_0 = s)$, the distribution of the state at time $k=0$. The term $\beta_k(s)$ can be computed via the recursion

$$\beta_k(s) = \sum_{s' \in \mathcal{S}} \beta_{k+1}(s') \gamma_k(s, s') \quad (10)$$

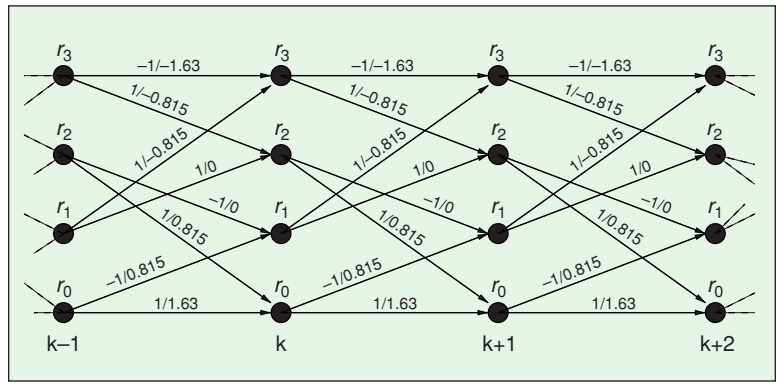
with the initial value $\beta_N(s) = 1$ for all $s \in \mathcal{S}$. The term $\gamma_k(s_k, s_{k+1})$ can be further decomposed into

$$\gamma_k(s_k, s_{k+1}) = P(s_{k+1}|s_k) \cdot p(y_k|s_k, s_{k+1}). \quad (11)$$

The transition probability $\gamma_k(r_i, r_j)$ is zero if the index pair (i, j) is not in \mathcal{B} . For pairs (i, j) from \mathcal{B} , the probability $P(s_{k+1}|s_k)$ is governed by the corresponding input symbol $x_{i,j}$ and $p(y_k|s_k, s_{k+1})$ is governed by the corresponding output symbol $v_{i,j}$, i.e.,

$$\gamma_k(r_i, r_j) = \begin{cases} P(x_k = x_{i,j}) \cdot p(y_k|v_k = v_{i,j}), & (i, j) \in \mathcal{B}, \\ 0, & (i, j) \notin \mathcal{B}. \end{cases} \quad (12)$$

For example, the term $\gamma_k(r_0, r_0)$ for the trellis in Figure 5 is $P(x_k = +1) \cdot p(y_k|v_k = 1.63)$, but the term $\gamma_k(r_0, r_3)$ is zero. From the channel law (1), i.e.



▲ 5. A trellis representation of the channel in Figure 4. The states $r_0 = (1, 1), r_1 = (-1, 1), r_2 = (1, -1), r_3 = (-1, -1)$ are the possible contents of the delay elements in Figure 4. The transitions from a state $s_k = r_i$ at time k to a state $s_{k+1} = r_j$ at time $k+1$ are labeled with the input/output pair $x_{i,j} / v_{i,j}$.

$y_k = v_k + n_k$, and from the noise distribution it follows that $p(y_k|v_k)$ is given by

$$p(y_k|v_k) = \exp(- (y_k - v_k)^2 / (2\sigma^2)) / \sqrt{2\pi\sigma^2}.$$

Next, we assemble the three probabilities $\alpha(s_k)$, $\gamma_k(s_k, s_{k+1})$, and $\beta(s_k)$ to obtain the APP $P(x_k = x|\mathbf{y})$. All we need to accomplish this task is to sum the branch APPs $P(s_k, s_{k+1}|\mathbf{y})$ over all branches that correspond to an input symbol $x_k = x$, i.e.,

$$P(x_k = x|\mathbf{y}) = \sum_{\forall (i, j) \in \mathcal{B}: x_{i,j} = x} P(s_k = r_i, s_{k+1} = r_j|\mathbf{y}). \quad (13)$$

For example, to compute the APP $P(x_k = +1|\mathbf{y})$ using the trellis in Figure 5, the branch APPs of the index pairs (00), (12), (20), and (32) have to be summed over. We may include the demapping operation and seek the conditional LLR $L(c_k|\mathbf{y})$ of the code bit c_k :

$$L(c_k|\mathbf{y}) = \ln \frac{P(c_k = 0|\mathbf{y})}{P(c_k = 1|\mathbf{y})} = \ln \frac{P(x_k = +1|\mathbf{y})}{P(x_k = -1|\mathbf{y})}.$$

From (8), (13), and the decomposition of $p(s_k, s_{k+1}, \mathbf{y})$ it follows that

$$L(c_k|\mathbf{y}) = \ln \frac{\sum_{\forall (i, j) \in \mathcal{B}: x_{i,j} = +1} \alpha_k(r_i) \cdot \gamma_k(r_i, r_j) \cdot \beta_{k+1}(r_j)}{\sum_{\forall (i, j) \in \mathcal{B}: x_{i,j} = -1} \alpha_k(r_i) \cdot \gamma_k(r_i, r_j) \cdot \beta_{k+1}(r_j)}.$$

Finally, the code bit estimates \hat{c}_k are computed from the sign of $L(c_k|\mathbf{y})$ as in (5).

Since the forward/backward algorithm is a basic building block of our turbo equalization setup, we will give a concise description in terms of matrix operations. For a trellis with a set of states \mathcal{S} let a set of matrices \mathbf{P}_k of dimensions $|\mathcal{S}| \times |\mathcal{S}|$ be defined as

The process of making hard decisions on the channel symbols actually destroys information pertaining to how likely each of the possible channel symbols might have been.

$$\{\mathbf{P}_k\}_{i,j} = \gamma_k(r_i, r_j), \quad (14)$$

where $\{\cdot\}_{i,j}$ denotes the entry of a matrix in the i th row and j th column. Moreover, let the matrices $\mathbf{A}(x)$ be defined for $x \in \{+1, -1\}$ as

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i, j) \text{ is a branch with } x_{i,j} = x, \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

For example, for the trellis in Figure 5 we find that

$$\mathbf{A}(+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{A}(-1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let the matrices $\mathbf{B}_k(x)$ be the componentwise product of $\mathbf{A}(x)$ and \mathbf{P}_k . We arrive at the equations in Table 1, which reflect the forward/backward algorithm derived in (9)–(13). In particular, the vectors \mathbf{f}_k and \mathbf{b}_k keep track of the quantities $\alpha_k(s)$ and $\beta_k(s)$. (In a practical implementation of the algorithm, a frequent renormalization of the vectors is indicated to avoid numerical underflow. That is, after each step in the recursion to compute \mathbf{f}_k and \mathbf{b}_k , both vectors should be multiplied with the sum of all $|\mathcal{S}|$ entries. Thus, after normalization, all entries in \mathbf{f}_k or \mathbf{b}_k should add up to one.) The algorithm stated in Table 1 is derived for the

Table 1. Equalization forward/backward algorithm.
<i>Input:</i> Matrices \mathbf{P}_k and $\mathbf{B}_k(x)$ for $k = 1, \dots, N$, length- $ \mathcal{S} $ column vectors \mathbf{f}_k and \mathbf{b}_k with \mathbf{f}_0 and \mathbf{b}_N initialized to 1 for all entries.
<i>Recursively compute:</i> $\mathbf{f}_k = \mathbf{P}_{k-1} \mathbf{f}_{k-1}$, $k = 1, 2, \dots, N$, $\mathbf{b}_k = \mathbf{P}_k^T \mathbf{b}_{k+1}$, $k = N-1, N-2, \dots, 1$,
<i>Output:</i> For $k = 1, \dots, N$ output the LLR $L(c_k \mathbf{y}) = \ln \frac{\mathbf{f}_k^T \mathbf{B}_k(+1) \mathbf{b}_{k+1}}{\mathbf{f}_k^T \mathbf{B}_k(-1) \mathbf{b}_{k+1}}.$

situation that the channel is not in any predefined starting or ending state and can be easily modified to include this information.

Linear Equalization

The computational complexity of the trellis-based approaches is determined by the number of trellis states, equal to 2^L , that grows exponentially with the number L of delay elements in the tapped delay line in Figure 4. This problem is exacerbated with higher-order signal alphabets. For example, if the symbols x_k are from an 8-ary alphabet, there are 8^L states in the trellis.

In contrast to MAP equalization, linear-filter-based approaches perform only simple operations on the received symbols, which are usually applied sequentially to a subset \mathbf{y}_k of M observed symbols y_k . Suppose this subset is given by $\mathbf{y}_k = (y_{k-4} \ y_{k-5} \ \dots \ y_{k+6})^T$, i.e., $M=11$. For the considered length-three channel in this article, the channel law (1) can be expressed in matrix form $\mathbf{y}_k = \tilde{\mathbf{H}} \mathbf{x}_k + \mathbf{n}_k$ as

$$\mathbf{y}_k = \begin{bmatrix} b_2 & b_1 & b_0 & 0 & 0 & \dots & 0 \\ 0 & b_2 & b_1 & b_0 & 0 & \dots & 0 \\ & & & \ddots & \ddots & \ddots & \\ 0 & 0 & \dots & 0 & b_2 & b_1 & b_0 \end{bmatrix} \cdot \begin{bmatrix} x_{k-6} \\ \vdots \\ x_{k+6} \end{bmatrix} + \begin{bmatrix} n_{k-4} \\ \vdots \\ n_{k+6} \end{bmatrix}, \quad (16)$$

where $\tilde{\mathbf{H}}$ is an $M \times (M+L)$ matrix. The symbols in \mathbf{y}_k depend on transmitted symbols in the interval $\{k-d, \dots, k+d\}$ for $d=6$. This explains the choice of the subset \mathbf{y}_k , which is used to compute estimates \hat{x}_k of the transmitted symbol x_k . We assume that x_k , n_k , or y_k are zero if k is outside the range $\{1, 2, \dots, N\}$. Any type of linear processing of \mathbf{y}_k to compute \hat{x}_k can be expressed with the linear (affine) function

$$\hat{x}_k = \mathbf{f}_k^T \mathbf{y}_k + b_k, \quad (17)$$

where the length M , vector \mathbf{f}_k , and the scalar b_k are real-valued parameters subject to optimization. Clearly, this processing model is similar to (1), i.e., there exists a tapped delay line model similar to that in Figure 4 implementing (17).

The linear model (16) immediately suggests multiplying \mathbf{y}_k with a vector \mathbf{f}_k that recovers x_k perfectly in the absence of noise. With noise present an estimate $\hat{x}_k = x_k + \mathbf{f}_k^T \mathbf{n}_k$ is obtained. This so-called zero-forcing (ZF) equalizer [18] suffers from “noise enhancement,” which can be severe if $\tilde{\mathbf{H}}$ is ill conditioned. This effect can be avoided using (linear) minimum mean square error (MMSE) estimation [28].

A linear MMSE estimator computes \hat{x}_k such that the mean squared error (MSE) $E(|x_k - \hat{x}_k|^2)$ between x_k and \hat{x}_k is minimized, where $E(\cdot)$ denotes expectation.

This is achieved by the linear model

$$\begin{aligned}\hat{x}_k &= E(x_k) + \mathbf{f}_k^T (\mathbf{y}_k - E(\mathbf{y}_k)) \\ \mathbf{f}_k &= \text{Cov}(\mathbf{y}_k, \mathbf{y}_k)^{-1} \text{Cov}(\mathbf{y}_k, x_k),\end{aligned}\quad (18)$$

which is not purely linear because of the bias terms $E(x_k)$ and $E(\mathbf{y}_k)$. The covariance $\text{Cov}(\mathbf{a}, \mathbf{b})$ of two column vectors is defined as $E(\mathbf{a}\mathbf{b}^T) - E(\mathbf{a})E(\mathbf{b}^T)$. Using (16) we can evaluate \mathbf{f}_k and $E(\mathbf{y}_k)$ from

$$\begin{aligned}\text{Cov}(\mathbf{y}_k, \mathbf{y}_k) &= \sigma^2 \mathbf{I} + \tilde{\mathbf{H}} \text{Cov}(\mathbf{x}_k, \mathbf{x}_k) \tilde{\mathbf{H}}^T, \\ \text{Cov}(\mathbf{y}_k, x_k) &= \tilde{\mathbf{H}} \text{Cov}(\mathbf{x}_k, x_k), \\ E(\mathbf{y}_k) &= \tilde{\mathbf{H}} E(x_k),\end{aligned}\quad (19)$$

where \mathbf{I} is the $M \times M$ identity matrix. The independence assumption on the symbols x_k yields that the covariance $\text{Cov}(x_k, x_{k'})$ between any two different symbols x_k and $x_{k'}$ vanishes. The covariance matrix $\text{Cov}(\mathbf{x}_k, \mathbf{x}_k)$ is therefore diagonal, and $\text{Cov}(\mathbf{x}_k, x_k)$ is equal to $\mathbf{u} \cdot \text{Cov}(x_k, x_k)$, where \mathbf{u} is a column vector of zeros with the seventh position set to one corresponding to the seventh element x_k of \mathbf{x}_k . The remaining statistics $E(x_k)$ and $\text{Cov}(x_k, x_k)$ are obtained from $P(x_k)$

$$\begin{aligned}E(x_k) &= \sum_{\forall x \in \{+1, -1\}} x \cdot P(x_k = x), \\ \text{Cov}(x_k, x_k) &= \sum_{\forall x \in \{+1, -1\}} |x - E(x_k)|^2 \cdot P(x_k = x).\end{aligned}\quad (20)$$

The estimates \hat{x}_k are usually not in the symbol alphabet $\{+1, -1\}$ and the decision whether $+1$ or -1 have been transmitted is usually based on the estimation error $e_k = x_k - \hat{x}_k$. The distribution $p(e_k)$ of this error, given the estimator (18), has zero mean and variance $\text{Cov}(e_k, e_k) = \text{Cov}(x_k, x_k) - \mathbf{f}_k^T \tilde{\mathbf{H}} \mathbf{u}$ [20]. Assuming furthermore that $p(e_k)$ is Gaussian yields

$$p(e_k) = \exp(-e_k^2 / (2 \text{Cov}(e_k, e_k))) / \sqrt{2\pi \text{Cov}(e_k, e_k)}.$$

The hard decision of \hat{x}_k should be the symbol $x \in \{+1, -1\}$ maximizing $p(e_k)$, which turns out to be the symbol x of closest distance $|x - \hat{x}_k|$ to \hat{x}_k .

Using the fairly standard IID assumption on the symbols x_k , we find that $E(x_k) = 0$, $\text{Cov}(x_k, x_k) = 1$ holds according to (20), and we arrive at the standard linear MMSE equalizer [18] according to (18):

$$\hat{\mathbf{x}} = \mathbf{f}_k^T \mathbf{y}_k, \quad \mathbf{f}_k = (\sigma^2 \mathbf{I} + \tilde{\mathbf{H}} \tilde{\mathbf{H}}^T)^{-1} \tilde{\mathbf{H}} \mathbf{u}.$$

It is also possible to (nonlinearly) process previous estimates to find the \hat{x}_k besides the linear processing of \mathbf{y}_k . Such an approach is also filter-based and called decision-feedback equalization (DFE) [18].

Soft Processing

The equalizer can often provide more information to the decoder than just the hard decisions (at the cost of additional storage, processing, and communication), such as probabilities that x_k takes on a particular value from $\{+1, -1\}$. The principle of using probabilities (soft information) rather than hard decisions is often referred to as “soft decoding.” This is shown in the second implementation of receiver B in Figure 1(c), by indicating that soft information $s(\cdot)$ is flowing along the edges of the flowgraph.

A natural choice for the soft information $s(x_k)$ about the transmitted symbols x_k are the APPs $P(x_k|\mathbf{y})$ or, similarly, the LLRs $L(c_k|\mathbf{y})$ (including demapping), which are a “side product” of the MAP symbol detector. Also, the (somewhat less complex) Viterbi equalizer may produce approximations of $L(c_k|\mathbf{y})$ using, e.g., the soft-output Viterbi algorithm [29].

For filter-based equalizers, extracting soft information $s(x_k)$ is more involved [8], [12]. A common approach is to assume that the estimation error, $e_k = \hat{x}_k - x_k$, is Gaussian distributed with PDF $p(e_k)$. This approach can apply to other equalization algorithms producing estimates \hat{x}_k as well.

Decoding

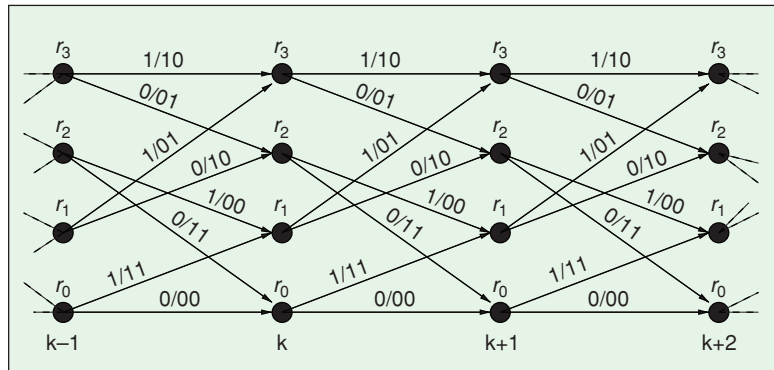
The LLRs $L(c_k|\mathbf{y})$ can be converted back to probabilities as

$$P(c_k = c|\mathbf{y}) = \frac{\exp(-c \cdot L(c_k|\mathbf{y}))}{1 + \exp(-L(c_k|\mathbf{y}))}, \quad (21)$$

where $c \in \{0, 1\}$. After deinterleaving $P(c_k|\mathbf{y})$ to $P(b_k|\mathbf{y})$, we are faced with the classical problem of decoding a binary convolutional code with probabilistic input. Let

$$\mathbf{p} = (P(b_1|\mathbf{y}) P(b_2|\mathbf{y}) \dots P(b_N|\mathbf{y}))^T$$

be the set of probabilities input to the decoder. We seek an efficient MAP decoder computing estimates \hat{a}_k of the transmitted data bits a_k from the LLRs $L(a_k|\mathbf{p})$, as in (5). Such a decoder may again use the forward/backward algorithm operating on a trellis



▲ 6. A trellis representation of the convolutional code of Figure 2. The trellis states correspond to the content of the delay elements as $r_0 = (0, 0)$, $r_1 = (1, 0)$, $r_2 = (0, 1)$ and $r_3 = (1, 1)$.

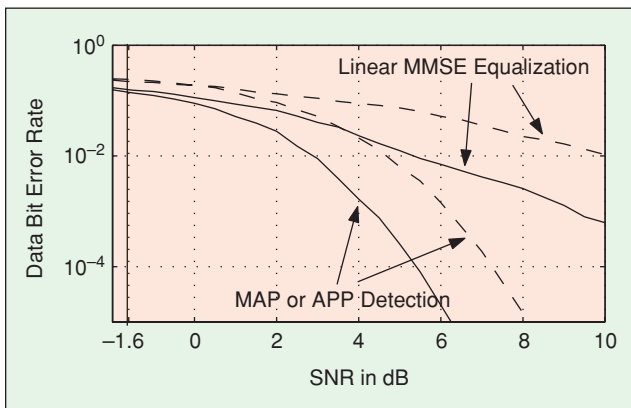
The process of only passing “extrinsic information” between constituent decoders is essential to the performance of turbo decoding algorithms.

description for the code, because its encoder in Figure 2 is a tapped delay line similar to that of the channel in Figure 4. A trellis description of the encoder is given in Figure 6. The trellis branches are denoted by the tuple $(i, j, a_{i,j}, b_{1,i,j}, b_{2,i,j})$, where $a_{i,j}$ is the input bit a_k and $(b_{1,i,j}, b_{2,i,j})$ are the two output bits (b_{2k-1}, b_{2k}) belonging to the state transition $(s_k = r_i, s_{k+1} = r_j)$. The set \mathcal{B} of valid transitions is the same as for the channel trellis in Figure 5.

To apply the forward/backward algorithm as in Table 1, we have to adjust the way in which the matrices \mathbf{P}_k and $\mathbf{A}(x)$ are formed. We start by redefining the transition probabilities $\gamma_k(s_k, s_{k+1})$ contained in \mathbf{P}_k

$$\gamma_k(r_i, r_j) = \begin{cases} P(a_k = a_{i,j}) \cdot \\ P(b_{2k-1} = b_{1,i,j} | \mathbf{y}) P(b_{2k} = b_{2,i,j} | \mathbf{y}), & (i, j) \in \mathcal{B}, \\ \dots & (i, j) \notin \mathcal{B}. \end{cases} \quad (22)$$

For example, the term $\gamma_k(r_0, r_0)$ for the trellis in Figure 6 equals $P(a_k = 0)P(b_{2k-1} = 0 | \mathbf{y})P(b_{2k} = 0 | \mathbf{y})$, where $P(a_k = 0) = 1/2$ under the IID assumption on the data bits. The code bit probabilities follow from (21). The matrices $\mathbf{A}(x)$ are defined for $x \in \{0, 1\}$ as



▲ 7. Performance of separate equalization and decoding (receiver B) when either hard estimates (dashed lines) or soft information (solid lines) are passed from the equalizer to the decoder. The systems transmits $K = 512$ data bits and uses a 16-random interleaver to scramble $N = 1,024$ code bits. The SNR threshold at -1.6 dB is the lowest SNR for which reliable transmission is possible.

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i, j) \text{ is a branch with } a_{i,j} = x, \\ 0, & \text{otherwise.} \end{cases}$$

Besides the LLRs $L(a_k | \mathbf{p})$ required to compute the estimates \hat{a}_k , the decoder may compute as well the LLRs $L(b_k | \mathbf{p})$. These LLRs will serve as a priori information for the equalizer forward/backward algorithm later. They can be computed using the forward/backward algorithm for decoding in Table 2 by simply changing the definition of the matrices $\mathbf{A}(x)$. The code bit LLRs $L(b_{2k-1} | \mathbf{p})$, $k = 1, 2, \dots, K$ are computed by choosing $\mathbf{A}(x)$ as

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i, j) \text{ is a branch with } b_{1,i,j} = x, \\ 0, & \text{otherwise.} \end{cases}$$

The LLRs $L(b_{2k} | \mathbf{p})$, $k = 1, 2, \dots, K$ are computed by choosing $\mathbf{A}(x)$ as

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i, j) \text{ is a branch with } b_{2,i,j} = x, \\ 0, & \text{otherwise.} \end{cases}$$

We arrive at the forward/backward algorithm for decoding in Table 2. We note that this algorithm uses a different initialization of the vectors \mathbf{f}_k and \mathbf{b}_k , which is due to the termination of the encoder to the zero state r_0 at time steps $k=0$ and $k=K$.

The performance of our example communication scheme, in conjunction with either hard bit estimates or soft information passed from the equalizer to the decoder (performing ML decoding), is depicted in Figure 7. We note the familiar 2-dB gain in SNR when soft information is used. Unfortunately, the best scheme (MAP symbol detection and decoding) is still 7.7 dB away from the performance limit at -1.6 dB SNR at a data error rate of 10^{-5} .

The unimpressive performance of the separate equalization and decoding strategies exhibited in Figure 7 is due to a number of independence assumptions in the derivation of the soft information exchanged. In particular, while computing the APPs $P(x_k | \mathbf{y})$ we invoked the IID assumption on the symbols x_k , i.e., all 2^N possible sequences \mathbf{x} are assumed to be equally likely, i.e., $P(\mathbf{x}) = 1/2^N$. There are only 2^K valid sequences \mathbf{x} , however, each belonging to a particular information word \mathbf{a} . The equalizer in receiver B should therefore compute the APPs as

$$P(x_k = x | \mathbf{y}) = \sum_{\text{all } 2^K \text{ valid } \mathbf{x}: x_k = x} \frac{p(\mathbf{y} | \mathbf{x}) P(\mathbf{x})}{p(\mathbf{y})}, \quad (23)$$

where $P(\mathbf{x}) = 1/2^K$ for valid \mathbf{x} . While this approach would yield significantly improved performance, the computational complexity is clearly prohibitive since we cannot use the trellis-based forward/backward algorithm anymore and must resort to exhaustive search.

Turbo Equalization

A closer look at the way in which the entries in \mathbf{P}_k are formed in (12) in the equalizer forward/backward

algorithm reveals that they consist of two entries, namely, $p(y_k|v_k=v_{i,j})$ and $P(x_k=x_{i,j})$, where $p(y_k|v_k=v_{i,j})$ can be interpreted as “local” evidence about which branch in the trellis was traversed, and $P(x_k=x_{i,j})$ takes on the role of “prior” information, which accounts for any prior knowledge about the probability of any branch in the trellis. In the separate equalization and decoding strategy given in the previous section, the equalizer does not have any prior knowledge available, and the formation of entries in \mathbf{P}_k relies solely on the observed data y_k . On the other hand, the decoder forms the corresponding entries in \mathbf{P}_k without any local observations but entirely based on bitwise probabilities $P(b_k|\mathbf{y})$ provided by the equalizer. The bitwise probabilities are assembled into prior probabilities on the branches in (22). In any case, the forward/backward algorithm can be abstracted as a device that takes any observation \mathbf{y} and any bitwise prior probability and produces bit-wise APP values. A block diagram building block of the forward backward algorithm is given in Figure 8.

The performance of a forward/backward algorithm can be greatly improved if good a priori information is available. Since we have two forward/backward algorithms, one for equalization and one for decoding, it is a natural idea to feed back the APP values of the individual bits obtained by one forward/backward algorithm as a priori information to the other. This is the main idea of turbo equalization. Some caution has to be exercised when creating the feedback loop, however. In particular, we have to avoid creating direct feedback that is too strong. This consideration leads to the notion of extrinsic and intrinsic information.

We showed already that the conditional LLR $L(a_k|\mathbf{y})$ of the bit a_k given the observation \mathbf{y} splits into the extrinsic LLR $L_{\text{ext}}(a_k|\mathbf{y})$ of a_k contained in \mathbf{y} plus the intrinsic LLR $L(a_k)$. It follows from (7) that $L_{\text{ext}}(a_k|\mathbf{y})$ does not depend on $L(a_k)$.

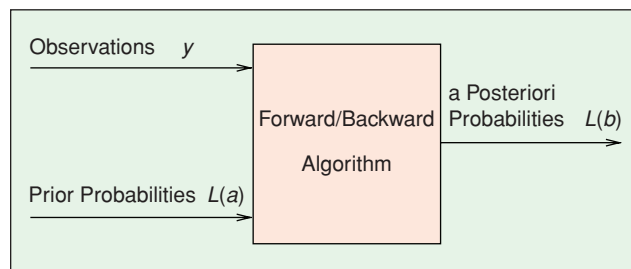
In the case of the forward/backward equalization algorithm that outputs $L(c_k|\mathbf{y})$, we can apply the same functional relation to separate the two contributions to $L(c_k|\mathbf{y})$ into extrinsic information $L_{\text{ext}}(c_k|\mathbf{y})=L(c_k|\mathbf{y})-L(c_k)$ and intrinsic information $L(c_k)$. In turbo equalization, as in the turbo decoding principle in general, only extrinsic information is fed back, as depicted in Figure 9. Feeding back intrinsic information $L(c_k)$ (included in $L(c_k|\mathbf{y})$) would create direct positive feedback, which would lead to fast convergence, though typically far from the globally optimal solution.

The interleavers are included into the iterative update loop to further disperse the direct feedback effect. In particular, the forward/backward algorithm creates output

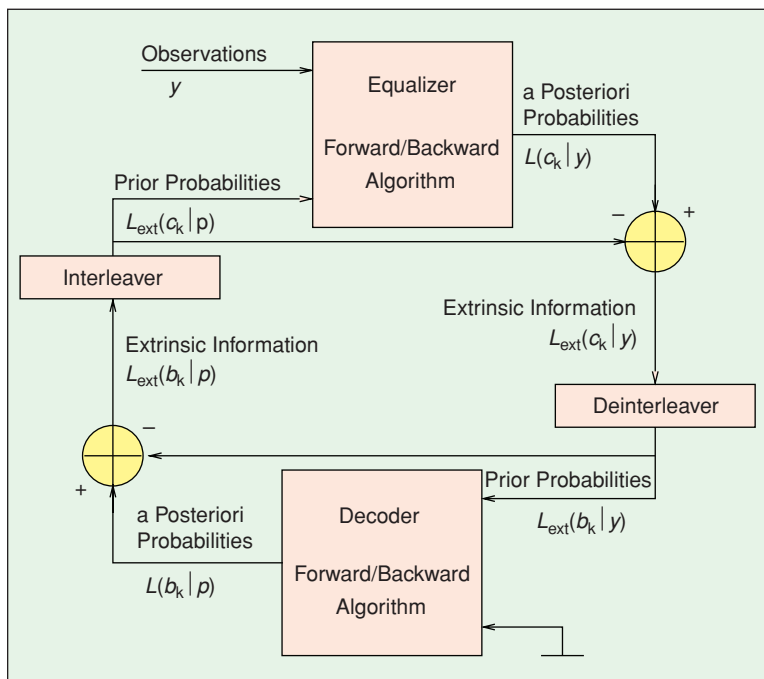
that is locally highly correlated. The correlations between neighboring symbols are largely suppressed by the use of an interleaver.

The notation of $L(\mathbf{b}|\mathbf{p}) = \text{Forward/Backward } (L_{\text{ext}}(\mathbf{b}|\mathbf{y}))$ will be used for the formation of an output sequence of LLRs $L(b_k|\mathbf{p})$ using the forward/backward algorithm for decoding as shown in Table 2 using the LLRs $L_{\text{ext}}(c_k|\mathbf{y})$ deinterleaved to $L_{\text{ext}}(b_k|\mathbf{y})$ as a priori LLRs. We will use the notation $L(\mathbf{c}|\mathbf{y}) = \text{Forward/Backward } (L_{\text{ext}}(\mathbf{c}|\mathbf{p}))$ for the formation of an output sequence of LLRs using the forward/backward algorithm for the equalizer utilizing the extrinsic LLRs $L_{\text{ext}}(b_k|\mathbf{p})$ interleaved to $L_{\text{ext}}(c_k|\mathbf{p})$ as a priori LLRs. We summarize the operations of a turbo equalization scheme in Table 3.

While this formulation of the turbo equalization algorithm is based on two forward/backward algorithms,



▲ 8. The forward/backward algorithm as a schematic block taking prior probabilities and observations as input and producing a posteriori probabilities as output.



▲ 9. Block diagram of a turbo equalizer. At the center of the turbo equalizer are two forward/backward algorithms that can operate on observations and prior information about individual bits. Only the extrinsic information is fed back in the iterative loop. The observation input of the forward backward algorithm for the decoder is grounded to indicate that the decoding algorithm operates on a priori values alone.

any pair of equalization and decoding algorithms that make use of soft information can be used as constituent algorithms for the turbo equalizer by simply replacing the operation $L(\mathbf{c}|\mathbf{y}) = \text{Forward/Backward}(L_{\text{ext}}(\mathbf{c}|\mathbf{p}))$ by another operation, $L(\mathbf{c}|\mathbf{y}) = \text{Equalizer}(L_{\text{ext}}(\mathbf{c}|\mathbf{p}))$, where this operation is any equalization algorithm that can map channel observations and soft inputs from the decoder into soft outputs to feed back into the decoder.

For example, the linear MMSE equalizer can take advantage of the a priori LLRs $L_{\text{ext}}(c_k|\mathbf{p})$ interpreted as probabilities on the transmitted symbol x_k by recomputing $E(x_k)$ and $\text{Cov}(x_k, x_k)$ according to (20), re-estimation of \hat{x}_k via (18), and recomputing of $s(x_k)$. Just as with the forward/backward equalization algorithm, to avoid short feedback cycles, such computations are typically undertaken under the constraint that $s(x_k)$ is not a function of $L_{\text{ext}}(c_k|\mathbf{p})$ at the same index k [12], [13]. This is equivalent to extracting only the extrinsic part of the information in the iterative scheme. We also note that several low-complexity alternatives for re-estimating \hat{x}_k exist [12]–[14], [30]–[32].

The performance of our communication scheme using receiver D from Figure 1 (turbo equalization) is depicted in Figures 10 and Figure 11. We see a significant performance gain over the iterations using either

Table 2. Decoding forward/backward algorithm.

Input: Matrices \mathbf{P}_k and $\mathbf{B}_k(x)$ for $k = 1, \dots, K$, length- $|\mathcal{S}|$ column vectors \mathbf{f}_k and \mathbf{b}_k with \mathbf{f}_0 and \mathbf{b}_K initialized to 0 for all entries except the first entry being 1.

Recursively compute:

$$\begin{aligned} \mathbf{f}_k &= \mathbf{P}_{k-1} \mathbf{f}_{k-1}, \quad k = 1, 2, \dots, K, \\ \mathbf{b}_k &= \mathbf{P}_k^T \mathbf{b}_{k+1}, \quad k = K-1, k-2, \dots, 1. \end{aligned}$$

Output: For $k = 1, 2, \dots, K$ output the LLR

$$L(a_k|\mathbf{p}) = \ln \frac{\mathbf{f}_k^T \mathbf{B}_k(0) \mathbf{b}_{k+1}}{\mathbf{f}_k^T \mathbf{B}_k(1) \mathbf{b}_{k+1}}.$$

The LLRs $L(b_k|\mathbf{p})$ are computed similarly.

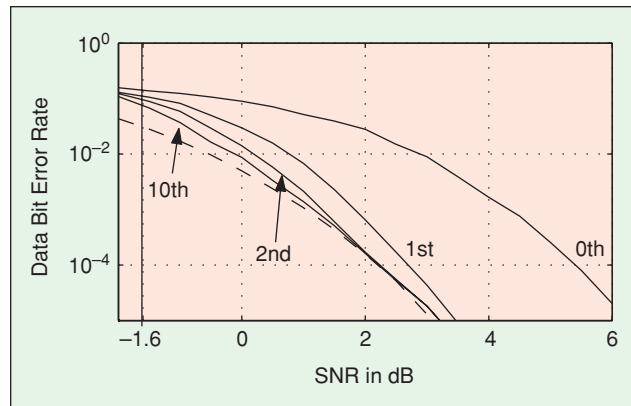
Table 3. Turbo equalization algorithm.

Input: Channel coefficients h_l for $l = 0, 1, \dots, L$.
Observation sequence \mathbf{y} .
A sequence of LLRs $L_{\text{ext}}(\mathbf{c}|\mathbf{p})$ initialized to 0.
A predetermined number of iterations ℓ .

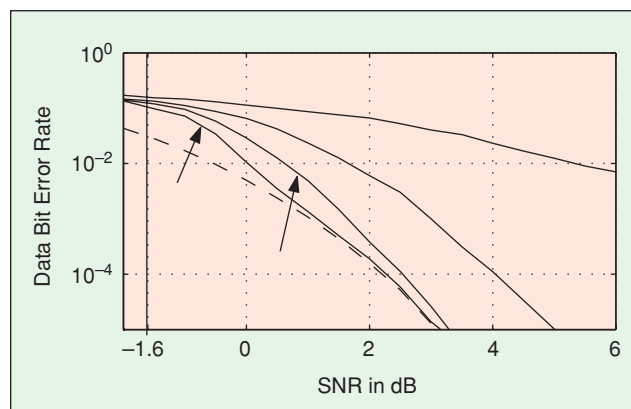
Recursively compute for ℓ iterations:

$$\begin{aligned} L(\mathbf{c}|\mathbf{y}) &= \text{Forward/Backward}(L_{\text{ext}}(\mathbf{c}|\mathbf{p})) \\ L_{\text{ext}}(\mathbf{c}|\mathbf{y}) &= L(\mathbf{c}|\mathbf{y}) - L_{\text{ext}}(\mathbf{c}|\mathbf{p}) \\ L(\mathbf{b}|\mathbf{p}) &= \text{Forward/Backward}(L_{\text{ext}}(\mathbf{b}|\mathbf{y})) \\ L_{\text{ext}}(\mathbf{b}|\mathbf{p}) &= L(\mathbf{b}|\mathbf{p}) - L_{\text{ext}}(\mathbf{b}|\mathbf{y}) \end{aligned}$$

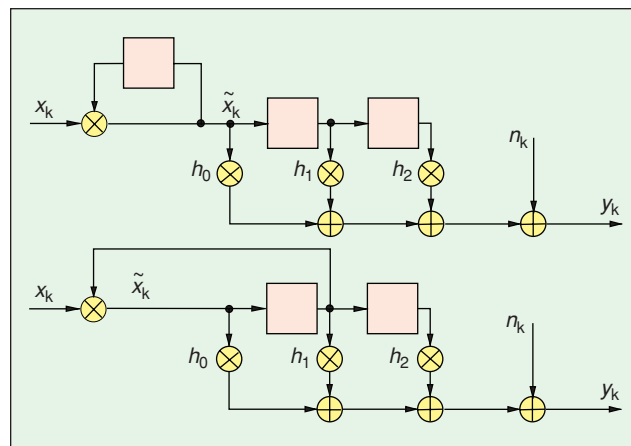
Output: Compute data bit estimates \hat{a}_k from $L(a_k|\mathbf{y})$.



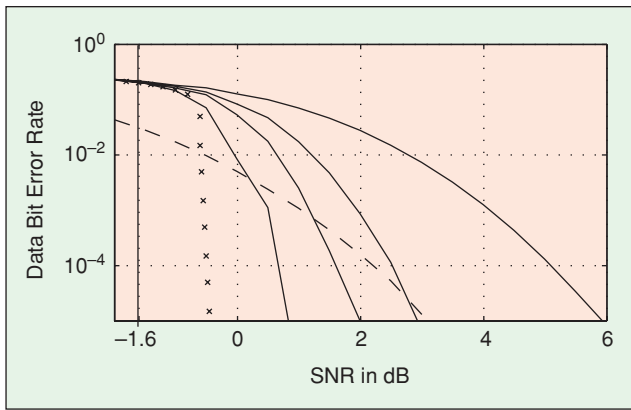
▲ 10. Performance of turbo equalization after 0, 1, 2, and 10 iterations using APP detection. The system parameters are as in Figure 7 ($K=512$, 16-random interleaver, SNR limit for reliable transmission: -1.6 dB).



▲ 11. Performance of turbo equalization after 0, 1, 2, and 10 iterations using linear MMSE equalization. The system parameters are as in Figure 7 ($K=512$, 16-random interleaver, SNR limit for reliable transmission: -1.6 dB).



▲ 12. Two equivalent tapped delay line circuits of the channel model (1) for $L=2$ including the precoder $\tilde{x}_k = x_k \cdot \tilde{x}_{k-1}$. That is, the symbols x_k are mapped before transmission to the symbols $\tilde{x}_k = x_k \cdot \tilde{x}_{k-1}$. This mapping will not cause a complexity overhead, since there exists a tapped delay circuit including the precoder still requiring only two delay elements.



▲ 13. Performance of turbo equalization after 0, 1, 2, and 10 iterations using MAP symbol detection and the precoder $\tilde{x}_k = x_k \cdot \tilde{x}_{k-1}$. The system parameters are as in Figure 7 ($K=512$, 16-random interleaver, SNR limit for reliable transmission: -1.6 dB). The line displayed with "x" marks is the performance of the same system with $K = 25,000$ and S random interleaving ($S = 40$) after 20 iterations.

the APP detector or the linear MMSE equalizer. We note that for larger block lengths K , the linear MMSE equalizer performance approaches that of the APP detector [12]. The performance is lower bounded by that of the underlying rate-1/2 code used over an ISI-free channel at the same SNR, however. To improve (lower) this bound and to approach the performance limit (we are still 6 dB away at 10^{-5} BER), we require a different error-correction code, such as the recursive inner precoder $\tilde{x}_k = x_k \cdot \tilde{x}_{k-1}$ [33], [34] added to the channel model as shown in Figure 12. Note the improvement in the lower bound to within 1 dB of the performance limit and the corresponding improvement in the turbo equalization algorithm in Figure 13.

Extensions and Concluding Remarks

We see that the turbo equalization approach to reliable communication over ISI channels is tightly connected to recent work in turbo codes, LDPC codes, and iterative decoding algorithms. Just as turbo equalization grew out of the original turbo decoding algorithm, there are numerous extensions of the basic turbo equalization approach in which a wide variety of signal processing tasks are incorporated into the joint estimation process [6]–[8], [10]–[14], [35]–[41]. There are a host of additional ways in which to explore the turbo equalization approach based on explicit graphical models and iterative estimation algorithms, and we refer the interested reader to the references mentioned here, the references contained therein and, of course, to the contributions contained in this special section. We hope that this brief overview of the general concepts and salient features enables further exploration into this topic that we find to be both challenging and fascinating.

Ralf Koetter received his diploma in electrical engineering from the TU Darmstadt, Germany, in 1990 and his

Ph.D. degree from the Department of Electrical Engineering, Linköping University, Sweden. From 1996 to 1997, he was a visiting scientist at the IBM Almaden Research Lab. in San Jose, California. He was a visiting assistant professor at the University of Illinois at Urbana/Champaign and a visiting scientist at CNRS in Sophia Antipolis, France from 1997 to 1998. He joined the faculty of the University of Illinois at Urbana/Champaign in 1999 and is currently an associate professor at the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering. His research interests include coding and information theory and their application to communication systems. He was associate editor for coding theory and techniques for *IEEE Transactions on Communications* and is an associate editor for coding theory of *IEEE Transactions on Information Theory*. He received an IBM Invention Achievement Award in 1997, an NSF Career Award in 2000, and an IBM Partnership Award in 2001.

Andrew C. Singer received the S.B., S.M., and Ph.D. degrees, all in electrical engineering and computer science, from the Massachusetts Institute of Technology in 1990, 1992, and 1996, respectively. Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently an associate professor and a research associate professor in the Coordinated Science Laboratory. During the academic year 1996, he was a postdoctoral research affiliate in the Research Laboratory of Electronics at MIT. From 1996 to 1998, he was a research scientist at Sanders, A Lockheed Martin Company, in Manchester, New Hampshire. His research interests include statistical signal processing and communication, universal prediction and data compression, and machine learning. He was a Hughes Aircraft Masters Fellow and the recipient of the Harold L. Hazen Memorial Award for excellence in teaching in 1991. In 2000, he received the National Science Foundation CAREER Award, in 2001 the Xerox Faculty Research Award, and in 2002 was named a Willett Faculty Scholar. He is an associate editor for *IEEE Transactions on Signal Processing* and is a member of the MIT Educational Council, Eta Kappa Nu, and Tau Beta Pi.

Michael Tüchler studied electrical engineering from 1995 to 1997 at the Technical University Ilmenau, Germany, and from 1997 to 1998 at the Technical University Munich, Germany. He received his M.S. degree from the University of Illinois at Urbana-Champaign in 2000. Currently he is pursuing a Ph.D. degree at the Institute for Communications Engineering at the Technical University Munich, Germany. In 2001, he was a visiting scientist at the IBM Research Lab Zurich, Switzerland. His research interests include coding theory, equalization, and estimation techniques with application to wireline and wireless communication systems, magnetic recording, and sensor networks.

References

- [1] F. Kschischang, B. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.
- [2] H.E. Gamal, "On the theory and application of space-time and graph-based codes," Ph.D. dissertation, Dept. Elec. Comput.Eng., Univ. of Maryland at College Park, 1999.
- [3] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Inst. Systems Theory, Linköping University, Linköping, Sweden, 1996.
- [4] R. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.
- [5] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Mag.*, vol. 21, pp. 28–41, Jan. 2004.
- [6] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenated trellis coded modulation with iterative decoding: Design and performance," in *Proc. IEEE Global Telecomm. Conf.*, Nov. 1997.
- [7] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.
- [8] X. Wang and H. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, 1999.
- [9] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo equalization," *European Trans. Telecomm.*, vol. 6, pp. 507–511, Sept.–Oct 1995.
- [10] G. Bauch and V. Franz, "A comparison of soft-in/soft-out algorithms for 'Turbo detection'," in *Proc. Int. Conf. Telecomm.*, June 1998, pp. 259–263.
- [11] A. Anastasopoulos and K. Chugg, "Iterative equalization/decoding for TCM for frequency-selective fading channels," in *Conf. Record 31th Asilomar Conf. Signals, Systems Comp.*, vol. 1, Monterey, CA, Nov. 1997, pp. 177–181.
- [12] M. Tüchler, R. Koetter, and A. Singer, "Turbo equalization: Principles and new results," *IEEE Trans. Comm.*, vol. 50, pp. 754–767, May 2002.
- [13] M. Tüchler, A. Singer, and R. Kötter, "Minimum mean squared error (MMSE) equalization using priors," *IEEE Trans. Signal Processing*, vol. 50, pp. 673–683, Mar. 2002.
- [14] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," in *Proc. Intern. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp. 96–102.
- [15] S. Lin and J.J. Costello, *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [16] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 1, pp. 366–386, Jan. 2001.
- [17] C. Heegard and S. Wicker, *Turbo Coding*. Boston, MA: Kluwer, 1999.
- [18] J. Proakis and M. Salehi, *Communication Systems Engineering*. Upper Saddle River, NJ: Prentice-Hall, 1994, pp. 577–595.
- [19] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
- [20] W. Hirt and J. Massey, "Capacity of the discrete-time Gaussian channel with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 34, pp. 380–388, May 1988.
- [21] D. Arnold and H.-A. Loeliger, "On the information rate of binary-input channels with memory," in *Proc. IEEE Intern. Conf. Comm.*, vol. 9, June 2001, pp. 2692–2695.
- [22] A. Kavčić, "On the capacity of Markov sources over noisy channels," in *Proc. Global Commun. Conf. (GlobeCom)*, Nov. 2003.
- [23] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo codes," *IEEE Trans. Comm.*, vol. 44, pp. 1261–1271, Oct. 1996.
- [24] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.
- [25] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 41, pp. 704–713, May 1995.
- [26] G. Forney, "Maximum-likelihood estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 18, pp. 363–378, May 1972.
- [27] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, pp. 257–286, Feb. 1989.
- [28] H. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. New York: Springer-Verlag, 1994.
- [29] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecomm. Conf.*, 1989, pp. 1680–1686.
- [30] M. Tüchler and J. Hagenauer, "Turbo equalization using frequency domain equalizers," in *Proc. Allerton Conf.*, Monticello, IL, Oct. 2000, pp. 1234–1243.
- [31] D. Raphaeli and A. Saguy, "Linear equalizers for Turbo equalization: A new optimization criterion for determining the equalizer taps," in *Proc. 2nd Intern. Symp. Turbo Codes*, Brest, France, Sept. 2000, pp. 371–374.
- [32] Z. Wu and J. Cioffi, "Turbo decision aided equalization for magnetic recording channels," in *Proc. Global Telecomm. Conf.*, Dec 1999, pp. 733–738.
- [33] K. Narayanan, "Effect of precoding on the convergence of turbo equalization for partial response channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 686–698, Apr. 2001.
- [34] I. Lee, "The effect of a precoder on serially concatenated coding systems with an ISI channel," *IEEE Trans. Comm.*, vol. 49, pp. 1168–1175, July 2001.
- [35] P. Alexander, A. Grant, and M. Reed, "Iterative detection and code-division multiple-access with error control coding," *European Trans. Telecomm.*, vol. 9, pp. 419–425, Sept.–Oct. 1998.
- [36] S. Ariyavisitakul and Y. Li, "Joint coding and decision feedback equalization for broadband wireless channels," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1670–1678, Dec. 1998.
- [37] A. Anastasopoulos and K. Chugg, "Adaptive soft-in soft-out algorithms for iterative detection with parameteric uncertainty," *IEEE Trans. Commun.*, vol. 48, pp. 1638–1649, Oct. 2000.
- [38] A. Anastasopoulos and K. Chugg, "Adaptive iterative detection for phase tracking in turbo-coded systems," *IEEE Trans. Commun.*, vol. 49, pp. 2135–2144, Dec. 2001.
- [39] J. Nelson, A. Singer, and R. Koetter, "Linear iterative turbo-equalization (LITE) for dual channels," in *Proc. 33rd Asilomar Conf. Signals, Systems, Comp.*, Oct. 1999.
- [40] J.R. Barry, A. Kavčić, S.W. McLaughlin, A. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Mag.*, vol. 21, pp. 89–102, Jan. 2004.
- [41] J.M.F. Moura, J. Lu, and H. Zhang, "Structured LDPC codes with large girth," *IEEE Signal Processing Mag.*, vol. 21, pp. 42–55, Jan. 2004.