Turbo equalization and an M-BCJR algorithm for strongly narrowband intersymbol interference

Anderson, John B; Prlja, Adnan

Published in:
[Host publication title missing]

DOI:
10.1109/ISITA.2010.5648949

2010

Turbo Equalization and an M-BCJR Algorithm for Strongly Narrowband Intersymbol Interference

John B. Anderson and Adnan Prlja
Electrical and Information Tech. Dept. and Strategic Center for High Speed Wireless Communication
Lund University, Box 118, SE-221 00 Lund SWEDEN
Email: anderson@eit.lth.se, adnan.prlja@eit.lth.se

***Abstract*—An M-BCJR algorithm is proposed and tested over an AWGN channel with moderate to very intense intersymbol interference (ISI). Two M-BCJR applications are evaluated, simple detection over the ISI channel and turbo equalization. The signaling is binary faster than Nyquist linear modulation. The ISI models tested correspond to transmission of increasingly many bits/Hz-s with a fixed signal spectra; the paper studies the range 2–8 bits/Hz-s, which implies ISI models as long as 32 taps. As a simple detector, the M-BCJR achieves the ML error rate with small computation, even when the Viterbi algorithm is completely impractical. In turbo equalization, the M-BCJR needs somewhat more computation and a more careful design because it must produce accurate likelihoods.**

I. Introduction

We investigate turbo equalization and a new M-algorithm BCJR (M-BCJR) for some cases of intersymbol interference (ISI) that occur in severely bandlimited transmission. Such a scheme reduces the ordinary BCJR computation by retaining only the M largest terms at each recursion stage. The signal transmission model is baseband linear modulation according to

$$s(t) = \sqrt{E_b/T} \sum_n a_n h(t - n\tau T), \tag{1}$$

where $\{a_n\}$ are binary ± 1 data, E_b is the symbol energy, $h(t)$ is a unit energy pulse, and τT is the symbol time ($\tau \leq 1$). The pulse $h(t)$ is much more narrowband than $1/2\tau T$ Hz, and consequently there is strong ISI. An additive white Gaussian noise (AWGN) channel follows. We will employ this Linear Modulation—AWGN Channel—M-BCJR Algorithm system in two ways, as the inner coder/decoder in turbo equalization [9] and by itself as an uncoded narrowband communication system (called "simple detection" in the sequel). The receiver consists of a matched filter, a sampler and a post filter, which together reduce the channel model to a minimum phase discrete-time convolution of $\{a_n\}$ with $\boldsymbol{v} = v_0, v_1, \ldots$, to whose outputs are added zero-mean IID Gaussians with variance $N_0/2$.

The ISI creates a long model $\boldsymbol{v}$. It has been known for some time that a critical requirement in this scenario is providing a minimum phase input sequence to the receiver processor, whether it is a Viterbi algorithm (VA) or the M-BCJR in this paper. A straightforward means to do this is a matched filter, followed by an all-pass filter that produces a maximum phase output, followed by reversing the output frame. A whitened matched filter approach is possible, but a simpler and more tractable approach discussed in [7] is used in this paper (see Section II). The min phase idea was used with M-algorithm detection of ISI in [1] and later by others. We studied its role in reducing the ordinary VA and BCJR to a minimum size in [6], [7]. We find here that it plays a crucial role as well with the M-BCJR used in turbo equalization.

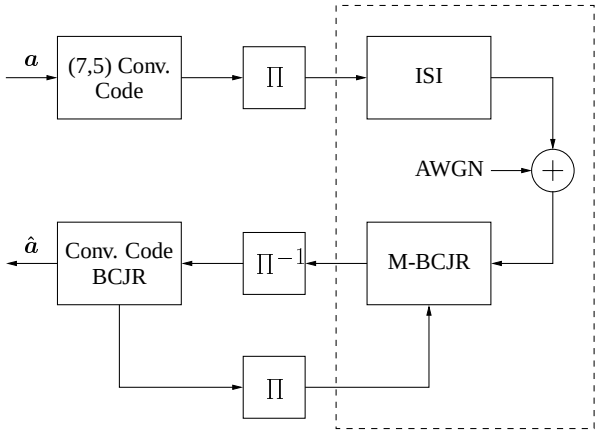

Fig. 1. Turbo equalization with a simple detection inner coder (dashed box).

Earlier work with reduced-state decoders primarily treats non-turbo receivers. A selection of recent papers on M- or other reduced BCJRs is [2], [3], [4], [5]. The BCJR procedure consists of forward and backward linear recursions, instead of the VA's unidirectional add-compare-select. The inner behavior of the BCJR can be rather different from the VA's. There is a major difference between an algorithm that calculates full log likelihoods (LLRs) and one that makes decisions about bits, i.e., calculates the LLR *sign*. This paper evaluates the M-BCJR as a simple detector where only the LLR sign matters and as part of turbo equalization, where it must produce accurate LLRs. These are shown schematically in Fig. 1. The M-BCJR algorithm presented is the outcome of experiments with many variations, but it follows closely the idea that both recursions should base their calculation on M major terms. It is not necessarily true that this is the best strategy for reducing the BCJR, and we will report some other strategies in a forthcoming paper.

The emphasis of this paper is strong bandwidth limitation,

chosen to place a severe strain on the state reduction design. In such a scenario, receiver error performance is sensitive to the entire shape of the signal spectrum, not just to a rough measure like 3 dB bandwidth. Signal sets with the same 3 dB bandwidth but different stopband spectrum shape can have quite different constrained capacities (see [8]). Removing only a small power from the outer spectrum can change the minimum distance of a set significantly; this is the "escaped distance" problem ([10], Chapter 6). These effects grow more pronounced as the bandwidth per data symbol drops. Our experimental method needs to be adjusted to this reality. If a small extra power appears in an outer stopband—for example, through a too-early truncation of a long $\boldsymbol{v}$—receiver bit error rate can improve, and give a false test result for that model.

With the foregoing in mind, the pulses $h(t)$ in this paper are chosen within the so-called faster than Nyquist, or FTN, framework. Throughout, we take $h(t)$ to be the root raised-cosine (RC) T-orthogonal pulse with 30% excess bandwidth. The spectrum of this pulse is in fact zero outside $\pm 1.3/2T$ Hz. $\tau = 1$ gives the familiar orthogonal pulse. As τ drops below 1, pulses come "faster", but the transmitted average power spectral density (PSD) shape remains the same, namely, a raised cosine. The bit density is $2/\tau$ data bits/Hz-s, taking the 3 dB bandwidth measure. The methods in this paper apply to any ISI, but there are good reasons for the FTN framework. First and foremost, the transmissions generated as τ declines yield increasingly high bit density systems that have an identical PSD shape. Second, they have proven to be an effective way to design coded systems that minimize both energy and bandwidth. Finally, the Shannon capacity of FTN signals exceeds that of orthogonal signals with the same PSD [8].

The paper is organized as follows. Section II presents the ISI channels with increasingly severe ISI. Section III reviews the BCJR algorithm, while Section IV describes an M-algorithm BCJR and evaluates the algorithm as a simple detector. Section V evaluates the algorithm as a component of turbo equalization.

II. Discrete-Time Channel Models

Our purpose in this section is to design a straightforward transmission system, in which signal spectra can be tightly controlled, PSDs equal to zero can be dealt with, and a minimum phase detection can be implemented. The inner part of the turbo equalization system, the dashed box in Fig. 1, consists of the following chain of processors:

Data $\{a_n\}$ at $n\tau T \rightarrow$ Linear modulation by $h(t)$ at rate $1/\tau T \rightarrow$ AWGN $\rightarrow$ Matched Filter $\rightarrow$ Sample at $n\tau T \rightarrow$ Discrete-time post filter $B(z) \rightarrow$ Frame reverse $\rightarrow$ M-BCJR $\rightarrow$ LLR Out

The chain produces a minimum phase sequence and applies it to the M-BCJR. The chain prior to the M-BCJR can be modeled as a discrete-time system that convolves $\{a_n\}$ with $\boldsymbol{v}$.

Next we will derive $\boldsymbol{v}$, using the Orthogonal Basis Method presented in [7]. There is no loss of generality if $h(t)$ in the above chain is replaced by $h(t) = \sum c_k \phi(t - k\tau T)$, where $c_k = \int h(t)\phi(t - k\tau T)\, dt$ and $\phi(t)$ is any convenient τT-orthogonal pulse that satisfies the sampling theorem for $h(t)$. Then the Matched Filter in the chain can be matched to $\phi(t)$, and since $\phi(t)$ is τT-orthogonal, the samples at $n\tau T$ are corrupted by AWGN. The $\{c_k\}$ are the energy-normalized samples $h(k\tau T)$ of the 30% root RC h, taken at the integers $k = -K, \ldots, K$. The filter $B(z)$ in the chain can be any allpass, and its outputs will again have AWGN.

Since the root RC pulse has zero PSD outside $1.3/2T$ Hz, it requires an infinite series expansion. We truncate the model $\boldsymbol{v}$ given to the M-BCJR, and the M-BCJR is thus slightly mismatched (model taps < 0.01 can be safely ignored). The model spectra in this paper are controlled by comparing their autocorrelation (from which their PSD can be derived). For each τ, the different models and the h samples are chosen long enough so that their autocorrelations agree to $\pm .001$.

The minimum phase requirement is implemented by a particular choice of the filter $B(z)$, namely the one whose poles are located at the zeros of the model $\boldsymbol{v}$ and whose zeros lie at the reflections about the unit circle of these positions. This actually produces a maximum phase output, but the Frame Reverse block produces the desired minimum phase. $B(z)$, however, can be *any* allpass, and there exist other B that improve M-BCJR performance even more than the min phase $B(z)$. The reason is that while min phase takes first priority, what the M-BCJR really needs is a model with a steep initial energy growth. The rise can be improved by allowing some low energy precursor taps in $\boldsymbol{v}$. The BCJR calculation needs to ignore the precursors, and it is thus slightly mismatched to the model, but the overall effect is a much smaller M for the same receiver performance. Such a modified min phase model is called super minimum phase [7].

A narrowband minimum phase model such as this thus has taps in the pattern [low energy precursor] + [high energy response] + [long decaying tail]. Some part of the precursor needs to be ignored in M-BCJR, as well as in other M-algorithm applications, since it increases their complexity exponentially with little improvement in the error rate or LLR quality. Ignoring the precursor is equivalent to running the decoder "ahead" by the length of the precursor. For the remainder of the response, branch labels ℓ at trellis stage n are generated from some ± 1 data sequence $\boldsymbol{a}$ by

$$\ell = \sum_{k=0}^{m} a_{n-k} v_k. \tag{2}$$

The memory m is the sum of the high energy and long tail lengths. Symbols $a_n, a_{n-1}, \ldots$ in the first term are the first high energy symbols. In narrowband ISI the long tail greatly increases the complexity of the full BCJR or VA [6], [7], but unlike the precursor, it has little effect on the M-BCJR.

We create a range of ISIs by choosing the FTN $\tau = 1/2, 1/3, 1/4$. The 1/2 case produces mild ISI and a 50%

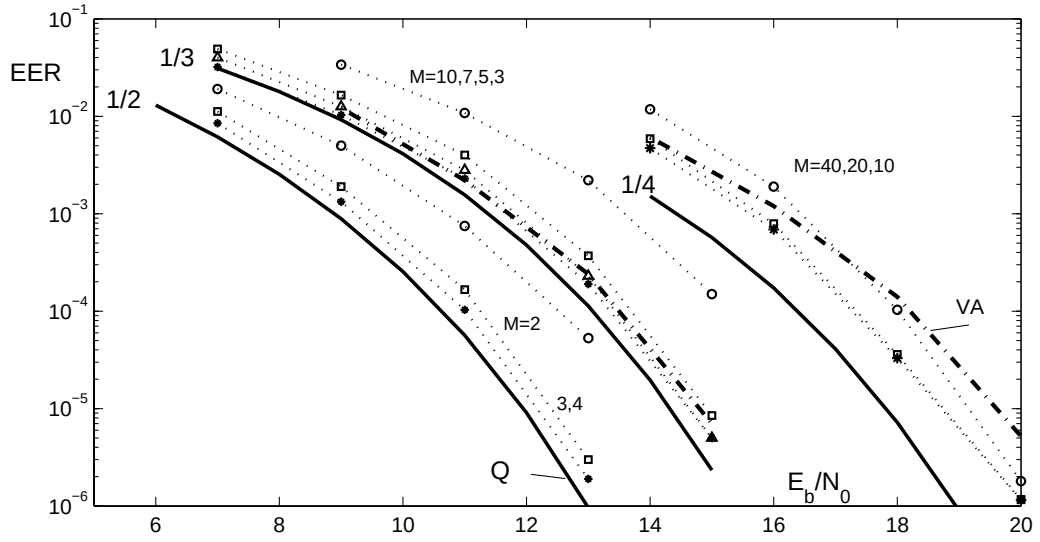

Fig. 2. Error event rates for simple ISI detection vs. E_b/N_0 in dB; M-BCJR (dotted), VA comparison (dashed). Heavy lines are Q-function estimates.

bandwidth reduction; a full BCJR or VA receiver for simple detection needs about 64 states. The 1/3 case is severe ISI and a 2/3 bandwidth reduction; a full BCJR/VA needs around 1000 states. The 1/4 case is very severe and a 3/4 reduction; a full BCJR/VA needs at least 262,000 states, and in fact only a reduced-state approach is practical. The signal sets created by these ISIs have square minimum distances of about 1.02, .58 and .20, which are energy losses of 2.9, 5.4 and 10 dB compared to antipodal signaling. $B(z)$ does not affect $d_{\min}$.

There is no framework at the moment for evaluating super min phase models, other than exhaustive testing. The following (3)–(5) are effective choices. Note that the original RC pulse train in (1) is not at all minimum phase. Additional tail taps are used in the generation of the actual transmitted sequence to insure spectral accuracy. Tap sets (6)–(7) are the true min phase sets for $\tau = 1/2, 1/3$. Set (6) is the first 18 taps of the min phase operation on 25 samples of the root RC pulse at $t = k\tau$, $k = -12, \ldots, 12$, and set (7) is the first 17 taps from 41 samples at $t = k\tau$, $k = -20, \ldots, 20$. It can be seen that the responses come sooner but energy growth is slower.

$$\boldsymbol{v} = \{-.005,-.003,.007,-.011,-.001,.034,-.019,.003, .375,.741,.499,-.070,-.214,.019,.087,-.020,-.028\} \quad (3)$$

$$\boldsymbol{v} = \{.025,.012,-.024,.008,.191,.464,.623,.506,.176, -.123,-.196,-.075,.060,.080,.013,-.035,-.022\} \quad (4)$$

$$\boldsymbol{v} = \{-.010,-.013,-.007,.005,.011,.004,-.008,.001,.060, .181,.339,.473,.520,.443,.262,.047,-.120,-.182, -.138,-.037,.055,.092,.070,.018,-.025,-.037,-.021, .003,.016,.012,.0004,-.008\} \quad (5)$$

$$\boldsymbol{v} = \{.177,.567,.694,.238,-.239,-.153,.123,.075,-.073, -.022,.040,-.007,-.016,.017,.001,-.014,.006,.005\} \quad (6)$$

$$\boldsymbol{v} = \{.038,.168,.385,.567,.549,.288,-.055,-.236,-.161, .032,.132,.071,-.040,-.070,-.014,.036,.025\} \quad (7)$$

In (3) there are 17 taps; the first 8 are taken as precursors and $m = 8$. Set (4) also has 17 taps, but only the first 4 are precursors and $m = 12$. Set (5) needs to be long and has 32 taps, 8 precursor taps and $m = 23$.

III. Review of the BCJR Algorithm

The BCJR computes the probabilities of states and paths in a signal trellis, given the channel outputs $\boldsymbol{y} = y_1, \ldots, y_N$ and the apriori data probabilities. The algorithm is given by two matrix recursions that calculate trellis working variables $\boldsymbol{\alpha}_n$ and $\boldsymbol{\beta}_n$ at stage n. These vectors have components

$$\begin{aligned} \alpha_n[j] &\triangleq P[\text{Observe } y_1, \ldots, y_n \cap \mathfrak{S}_n = j] \\ \beta_n[i] &\triangleq P[\text{Observe } y_{n+1}, \ldots, y_N \mid \mathfrak{S}_n = i] \end{aligned} \quad (8)$$

where $\mathfrak{S}_n$ is the encoder state at time n. The following *forward* and *backward recursions* hold:

$$\begin{aligned} \boldsymbol{\alpha}_n &= \boldsymbol{\alpha}_{n-1}\boldsymbol{\Gamma}_n, \quad n = 1, \ldots, N \\ \boldsymbol{\beta}_n &= \boldsymbol{\Gamma}_{n+1}\boldsymbol{\beta}_{n+1}, \quad n = N-1, \ldots, 1 \end{aligned} \quad (9)$$

Here $\boldsymbol{\Gamma}_n$ is the matrix with $[i, j]$ element

$$\begin{aligned} \Gamma_n[i,j] &\triangleq P\left[y_n \cap \mathfrak{S}_n = j \mid \mathfrak{S}_{n-1} = i\right] \\ &= \left[P(a')/\sqrt{\pi N_0/E_s}\right] \exp\left[-(N_0/E_s)(y_n - \ell_{i,j})^2\right] \end{aligned} \quad (10)$$

where $\ell_{i,j}$ is the label (2) on the branch from state i to j and a' is the value of data symbol a_n that causes the transition. Our data frames terminate at both ends at the all-0 state, so $\boldsymbol{\alpha}_0 = (1, 0, \ldots, 0)$ and $\boldsymbol{\beta}_N = (1, 0, \ldots, 0)'$. In a reduced recursion, some of the smaller $\boldsymbol{\alpha}_n$ or $\boldsymbol{\beta}_n$ components are set to 0 and have no further effect on the recursion.

The product of the $\{\boldsymbol{\alpha}\}$ and $\{\boldsymbol{\beta}\}$ produce the set $\{\boldsymbol{\lambda}\}$ through $\lambda_n[j] = \alpha_n[j]\beta_n[j]$, and from these come the LLRs via

$$\mathrm{LLR}(a_n) \triangleq \ln \frac{P[a_n = +1]}{P[a_n = -1]} = \ln \frac{\sum_{j \in \mathcal{L}_{+1}} \lambda_n[j]}{\sum_{j \in \mathcal{L}_{-1}} \lambda_n[j]} \quad (11)$$

Here $\mathcal{L}_{\pm 1}$ are the sets of states reached by $a_n = \pm 1$, for which nonzero α and β have both been found. A problem in a reduced calculation is that one or both of $\mathcal{L}_{\pm 1}$ can be *empty*. Both being empty at a reasonable M never occurred in our M-BCJR tests. If only one is empty, the numerator or denominator of (11) must be filled in by some backup method.

IV. An M-BCJR Algorithm and Its Simple Decoding Performance

The M-algorithm for searching code trees and trellises is well known, and we will only summarize it here. As a general procedure, the algorithm proceeds breadth-first through a tree structure of values, keeping only the dominant M at each tree stage. As a BCJR, we use it once each to find the dominant M α_n and β_n, near to the values that a full BCJR would find at n if it were applied to an ISI tree with a priori information. For moderate or strong ISI, the Γ-matrices are very sparse, and furthermore, most non-zero components are very small. A useful view is that the M-search implements a sparse matrix calculation in which the α or β vector at each stage is limited to M active components.

Our algorithm is as follows. Recursions start and end at state 0 (all +1s data). Inputs to the algorithm are the noisy channel outputs and *a priori* probabilities of the data. Outputs are the signed LLR values in (11). The M-list consists of two sublists, one containing α or β values and one containing the corresponding trellis states.

The α Recursion. Starting at $n = 0$, perform at stage $1, 2, \ldots$:

1). The α recursion in (9) is computed from the M nonzero values retained in $\boldsymbol{\alpha}_{n-1}$. There are M corresponding to data +1 and M to -1; only the $2M$ corresponding Γ elements are computed.

2). Trellis paths in the +1 and -1 M-lists may merge. Merges are detected and removed, leaving one survivor only, whose α value is the sum of the two incoming values.

3). The best M of the remaining paths are found. These are stored for the next iteration and for the β recursion.

The β Recursion. Starting at L, the end of the channel block, perform at stage $L, L-1, \ldots$:

4). The β recursion in (9) is computed from the M nonzero values retained in $\boldsymbol{\beta}_{n+1}$. There are M corresponding to data +1 and M to -1; only the $2M$ corresponding Γ elements are computed.

5). Trellis paths in the +1 and -1 M-lists may merge. Merges are detected and removed, leaving one survivor only, whose β value is the sum of the two incoming values.

6). The best M of the remaining paths are found, subject to the following condition: β paths *must be kept if their state and stage overlap with that of a stored* α.

7). Compute the LLR from (11). If $\mathcal{L}_{+1}$ or $\mathcal{L}_{-1}$ is empty, the LLR is $\pm\epsilon$, respectively, where ϵ is a threshold chosen in advance.

Notes on the algorithm operation. The overlap in step 6 never failed to occur in the tests presented here. Steps 3 and 6, which find the best M, are equivalent to finding the median of the larger list. An important property of median finding is that its computation is linear in M. This is because it never orders the elements, which requires order $M \log M$. In keeping with this, we take a true M-algorithm application to be one where *all* computation is of order M. The search for the median is thus implemented in order M, but so also is removal of state merges in steps 2 and 5 and finding the overlap of α and β in step 6. The key to the last two is keeping all path lists in state order, which is itself a linear operation. Finally, M need not be the same in the α and β recursion, but we found no significant gain from different M.

Figure 2 plots the error event rate of this M-BCJR algorithm used as a simple detector at the three ISI intensities, using taps (3)–(5). The algorithm decides symbols from the sign of its LLR output. Heavy lines show Q-function bounds that are the sum of several nearest error event terms weighted by their multiplicity factors (details are in [7]). Dashed heavy lines compare the performance at $\tau = 1/3$ and $1/4$ of a 256- and 4096-state VA, respectively. The M-BCJR performs better than the VA, especially at $\tau = 1/4$, because a practical VA cannot be large enough to deal with every detail of intense ISI. The M-BCJR needs only $M = 4, 7, 20$ at high E_b/N_0 for the three τ.

We will turn now to turbo decoding in which the M-BCJR is used as the ISI decoding element. Two M-BCJRs will be employed there. The first, called the "Simple" M-BCJR, is the one above. The second, called the "Backup" M-BCJR, has a more sophisticated step 7 that does not need an ϵ specified in advance. The ϵ plays no role in simple detection, because only the sign of the LLR matters, but in turbo decoding the best ϵ depends on SNR.

The new step 7 is as follows. First a third backup recursion is performed, which computes a symbol probability from the αs only. This works from the decided symbol path, which is available from the first two recursions. That path is traced forward through the signal trellis, and the "incorrect subsets" of each decided node are traced forward a certain length of stages. These traces are performed with a small M-search ($M = 2$ works well), and the necessary searching can be arranged in a simple way. The αs from this search give a backup estimate of $P[a_n = +1]/P[a_n = -1]$. This is used when $\mathcal{L}_{+1}$ or $\mathcal{L}_{-1}$ is empty; otherwise (11) is used.

V. M-BCJR in Turbo Equalization

In this section we evaluate the BER performance of the M-BCJR algorithm when applied as part of a turbo equalization system. We also show the advantages of super minimum phase discrete-time channel models (3) and (4) over the true minimum phase models (6) and (7) when the receiver is a BCJR with a truncated ISI reponse. The truncated BCJR calculates its labels based on the $m_{\rm trunc} + 1$ dominant taps, ignoring the low energy precursors in models (3) and (4). This

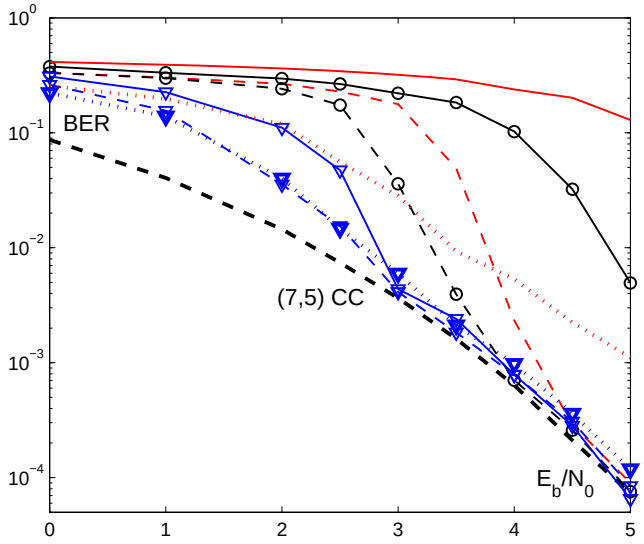

Fig. 3. Turbo equalization for $\tau = 1/2$, comparing Backup M-BCJR (dashed) with Simple M-BCJR (solid) and truncated BCJR (dotted) for $M = 4, 6, 16$ (respectively no markers, circles, triangles).

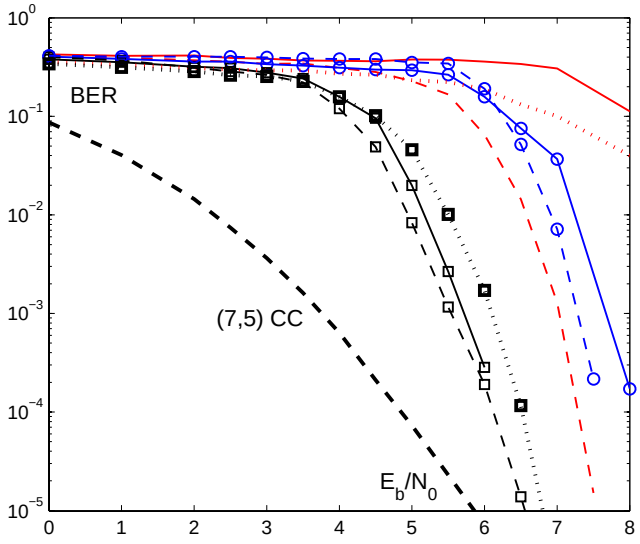

Fig. 4. Turbo equalization for $\tau = 1/3$; Simple, Backup and truncated as in Fig. 3. $M = 16, 64$ with no markers and squares; circles denote $A/B = 64/8$.

simple reduced-state BCJR will also serve as a comparison to the M-BCJRs from Section IV.

At the transmitter a block of N information bits is encoded by the (7,5) rate 1/2 convolutional code, producing a coded sequence of $2N$ symbols. These enter a size $2N$ random interleaver and map to antipodal symbols ± 1 ($0 = +1, 1 = -1$). After correct termination the transmitted signal is finally formed by (1) and tested over the simulated AWGN channel.

At the receiver iterative decoding is performed via turbo equalization where the component decoder for the intentional ISI is the M-BCJR (see Fig. 1). The block N is 1000 information bits and plots are based on ≥ 50 error events. ISI models are (3) and (4) for $\tau = 1/2$ and $1/3$. Soft information in terms of LLRs is passed around the turbo loop 10 times before a decision is made.

The output required from the M-BCJR differs now significantly from what is needed under simple decoding. Whereas only the sign of the LLRs was needed for data symbol decisions, turbo decoding requires reasonably accurate absolute values, especially in the early iterations.

Since no recursive precoding is employed, the ultimate turbo goal is to reach the performance of the underlying convolutional code, which is plotted as a reference. Figures 3–4 plot BER results for the Backup M-BCJR in Section IV (shown dashed), the Simple M-BCJR (solid), and the truncated BCJR (dotted). Figure 3 shows the results for ISI case (3). The three sets of curves correspond to different state space sizes: No marker corresponds to $M = 4$, circles to 6 and triangles to 16 states. It is clear that this ISI, which comes with a 50% bandwidth reduction, is not a major difficulty for either M-BCJR. The Backup M-BCJR clearly improves the BER performance over the Simple M-BCJR and as E_b/N_0 grows the performance becomes virtually that of the outer code. The truncated BCJR with only 4 states is not able to converge to the convolutional code due to the energy loss in the truncation.

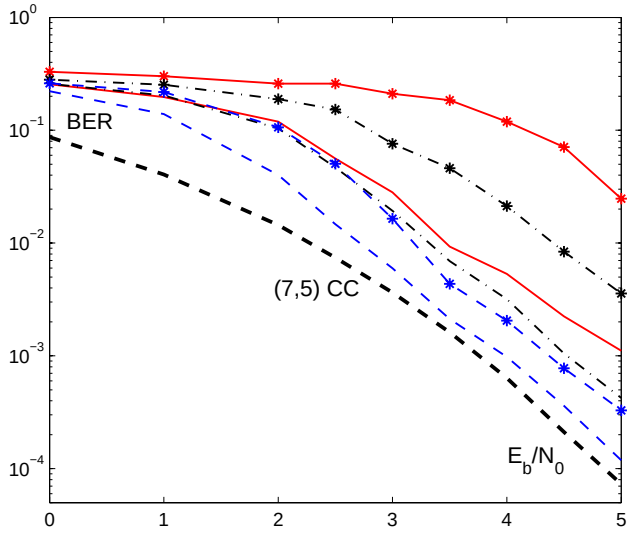

Fig. 5. BER performance of the truncated BCJR for $\tau = 1/2$ discrete-time channel models (3) (no markers) and (6) ($\star$) for different m_{trunc}.

Figure 4 plots the $\tau = 1/3$ case, which corresponds to much more severe ISI. The curves for $M = 16$ (no markers) and $M = 64$ (squares) show that the Backup M-BCJR design greatly improves the BER performance, and the convolutional code BER is very nearly achieved. The notation A/B denotes that the first turbo iteration is performed with $M = A$ and the remaining ones with $M = B$. The set of curves with circles has $A = 64$ and $B = 8$; it appears to represent a more effective use of computation.

We have performed an initial study of the extreme ISI case with $\tau = 1/4$ and the 32-tap set (5). Here the convolutional

code BER cannot be reached since it would violate capacity. BER is generally 3–4 dB worse than the $\tau = 1/3$ performance shown in Fig. 3. The M in the Backup M-BCJR needs to be in the range 50–100, compared to 20 in the simple detection case of Fig. 2 and many thousands for truncated-BCJR turbo decoding. The block length needs to be at least 4000 with this longer ISI. In addition, tests show that there need to be optimized scaling coefficients for the LLRs before the M-BCJR and the convolutional code BCJR in the turbo loop.

A comparison between the $\tau = 1/2$ discrete-time channel models (3) (super min phase) and (6) (ordinary min phase) is shown in Fig. 5. Here we use the truncated BCJR with three different m_{trunc} corresponding to 4, 8 and 16 states. The steeper initial energy growth clearly results in improved BER performance of the truncated BCJR. The same behavior has been observed for the $\tau = 1/3$ tap set.

VI. Conclusion

We have investigated several BCJR algorithms whose calculation is limited to M significant terms. As a simple detector, in a receiver with the correct overall minimum phase design, the M-BCJR hugely reduces computation under moderate and intense ISI in this paper. As part of a turbo equalizer, the application requires some accuracy in the LLR magnitudes, but the algorithm gives significant savings. It is important in both applications to convert to minimum phase before the M-BCJR and to allocate properly the precursor and main model components. Even better performance comes from sharpening the model energy by the super minimum phase method. The outcome is a turbo decoder of reasonable complexity, which can lead simultaneously to an energy saving of 4 dB and a bandwidth reduction of 35%.

Acknowledgments

This work was supported by the Swedish Research Council (VR) through Grant 621-2003-3210, and by the Swedish Foundation for Strategic Research (SSF) through its Strategic Center for High Speed Wireless Communication at Lund.

References

[1] K. Balachandran, J.B. Anderson, "Reduced complexity sequence detection for nonminimum phase intersymbol interference channels," *IEEE Trans. Information Theory*, vol. 43, pp. 275–280, Jan. 1997.

[2] C. Fragouli, N. Seshadri, W. Turin, "Reduced-trellis equalization using the BCJR algorithm," *Wireless Commun. & Mobile Computing*, vol. 1, pp. 397–406, 2001.

[3] G. Colavolpe, G. Ferrari, R. Raheli, "Reduced-state BCJR type algorithms," *IEEE J. Sel. Areas Communs.*, vol. 19, pp. 848–859, May 2001.

[4] D. Fertonani, A. Barbieri, G. Colavolpe, "Reduced-complexity BCJR algorithm for turbo equalization," *IEEE Trans. Communs.*, vol. 55, pp. 2279–2287, Dec. 2007.

[5] M. Sikora, D. J. Costello, Jr., "A new {SISO} algorithm with application to turbo equalization," *Proc.*, IEEE Int. Symp. Information Theory, Adelaide, Australia, 2005.

[6] A. Prlja, J.B. Anderson, F. Rusek, "Receivers for faster-than-Nyquist signaling with and without turbo equalization," *Proc.*, 2008 IEEE Int. Symp. Information Theory, Toronto, July 2008.

[7] J.B. Anderson, A. Prlja, F. Rusek, "New reduced state space BCJR algorithms for the ISI channel," *Proc.*, IEEE Int. Symp. Information Theory, Seoul, June 2009.

[8] F. Rusek, J.B. Anderson, "Constrained capacities for faster-than-Nyquist signaling," *IEEE Trans. Information Theory*, vol. 55, pp. 764–775, Feb. 2009.

[9] C. Douillard *et al.*, "Iterative correction of intersymbol interference: Turbo equalization," *Eur. Trans. Telecomm.*, vol. 6, pp. 507–511, Sept./Oct. 1995.

[10] J.B. Anderson, A. Svensson, *Coded Modulation Systems*, Kluwer-Plenum, New York, 2003.