Design and Implementation of M-BCJR algorithm based Turbo Equalizer

Priyatamkumar,[1] R.M.Banakar[1] and Shankaranand[2].
[1]*ENC Dept.,BVBCET,Hubli.India*
[2]*ENC Dept, NITK, Surathkal .India*
pk_bvb@yahoo.co.in, banakar@bvb.edu

Abstract

Turbo equalizers have been shown to be successful in mitigating the effects of inter-symbol interference introduced by partial response modems and by dispersive channels for code rates of R >1/2. We analyze the performance of iterative equalization and decoding (IED) using an M-BCJR equalizer. We use bit error rate (BER), frame error rate simulations and extrinsic information transfer (EXIT) charts to study and compare the performances of M-BCJR and BCJR equalizers on precoded and non-precoded channels. Using EXIT charts, the achievable channel capacities with IED using the BCJR, M-BCJR and MMSE LE equalizers are also compared. We predict the BER performance of IED using the M-BCJR equalizer from EXIT charts and explain the discrepancy between the observed and predicted performances by showing that the extrinsic outputs of the M-BCJR algorithm are not true logarithmic-likelihood ratios (LLR's). We show that the true LLR's can be estimated if the conditional distributions of the extrinsic outputs are known and finally we design a practical estimator for computing the true LLR's from the extrinsic outputs of the M-BCJR equalizer.

Keywords: Equalizer, ISI, EXIT, LLR's, extrinsic, intrinsic, apriori, aposteri.

1. Introduction

Turbo equalization (TEQ) [1] was proposed by Douillard *et al.* in 1995 for a rate R=1/2 convolutional-coded binary phase-shift keying (BPSK) system. Specifically, Douillard *et al.* demonstrated that the turbo equalizer was capable of mitigating the effects of inter-symbol interference (ISI), provided that the channel impulse response (CIR) was known. Here the performance improvements were obtained by performing the channel equalization and channel decoding iteratively. Gertsman and Lodge [2] then showed that the turbo principle could compensate for the performance degradations due to imperfect CIR estimation. Different iteration termination criteria [3], such as cross-entropy minimization [4], were also investigated in order to minimize the number of iterations carried out by the turbo equalizer. A turbo equaliza114
tion scheme was proposed by Bauch and Franz [5] for the global system for mobile communications (GSM) where different approaches of overcoming the dispersion of the so-called *a priori* information due to the interburst interleaving scheme were investigated. Research into combined R=1/3 convolutional turbo coding and iterative channel equalization has also been conducted by Raphaeli and Zarai [8]. Their results showed that for BPSK systems transmitting over nonrecursive channels the turbo equalizer using turbo codes outperformed the turbo equalizer utilizing convolutional codes. In the context of recursive channels, such as precoded magnetic storage media [9], the same trend was observed in the "floor" region of the bit-error rate (BER) curve. However, in the "cliff" region, the opposite trend was observed, where the turbo equalizer employing convolutional codes outperformed the turbo equalizer using turbo codes. With the ever-increasing demand for bandwidth, current systems aim to increase the spectral efficiency by invoking high-rate codes. This has been the motivation for research into block-turbo codes, which have been shown by Hagenauer *et al.* to outperform convolutional turbo codes using punctured high-rate convolutional component codes, when the coding rate is higher than R=2/3 . It was also observed that a rate L=.1024 block turbo code using BPSK over the nondispersive Gaussian channel can operate within 0.27 dB of the Shannon limit [11]. Another method of generating high-rate turbo codes have been proposed by Açikel and Ryan, whereby a rate R= 1/3 turbo code consisting of two R=1/2 convolutional codes is punctured. These high-rate turbo codes have been shown to perform better than the turbo codes proposed by Hagenauer *et al.* [10]. The

0-7803-8680-9/04/$20.00 ©2004 IEEE.

puncturing patterns were optimized for transmission over the nondispersive additive white Gaussian noise (AWGN) channel.

In this contribution, our objective is to investigate the performance of BPSK turbo equalizers employing different classes of high-rate codes, namely R=3/4 and R=5/6 convolutional codes, convolutional turbo codes, and block—turbo codes. This is because known turbo equalization results have only been presented for turbo equalizers using convolutional codes and convolutional turbo codes for code rates of R=1/3 and R=1/2 [1].

2. TURBO EQUALIZER

2.1 Principle of Turbo equalizer

Turbo equalizer is based on the principle of iterative decoding applied Turbo equalizer to SCCC's. The equalizer computes the *aposteriori* probabilities (APP's),$P(x_k = x|z_1, z_2, \ldots, z_K)$, $x \in X$, $k = 1, 2, \ldots, K$, given K received symbols z_k, $k = 1, 2, \ldots, K$ and outputs the extrinsic LLR's, $L^I(x_k)$, $k = 1, 2, \ldots, K$ defined as the *a posteriori* LLR minus the *a priori* LLR. The superscript *I* refer to the inner SISO module *viz.*, the equalizer.

$$L^I(x_k) = ln\left(\frac{P(x_k = +1/z_1, z_2, \ldots, z_k)}{P(x_k = -1/z_1, z_2, \ldots, z_k)}\right) - ln\left(\frac{P(x_k = +1)}{P(x_k = -1)}\right) \quad 2.1$$

The *a priori* LLR, $L(x_k)$, $k = 1, 2, \ldots, K$ represents the *a priori* information about the probability that $x_k \in X$, $k = 1, 2, \ldots, K$ assumes a particular value. The APP decoder provides them. In the first equalization step, the *a priori* information, $L(x_k)$, $k = 1, 2, \ldots, K$ is not available and all values are assumed to be equally probable i.e., $L(x_k) = 0$, $\forall k$. The *a priori* information for the decoder is obtained by deinterleaving the output extrinsic LLR's of the equalizer, $L(c_k) = \Pi^{-1}(L^I(x_k))$. Similar to the equalizer, the decoder also computes the APP's $P(c_k = c|L(c_1), L(c_2), \ldots, L(c_K))$, $c \in \{0, 1\}$ given the K code bit LLR's $L(c_k)$, $k = 1, 2, \ldots, K$ and outputs the extrinsic LLR's, $L^O(c_k)$, $k = 1, 2, \ldots, K$ defined as the output LLR minus the *a priori* LLR. The superscipt *O* refers to the outer decoder.

$$L^O(x_k) \quad ln\left(\frac{P(c_k) = +1 / L(c_1), L(c_2) \ldots L(c_k))}{P(c_k) = -1 / L(c_1), L(c_2), \ldots, L(c_k)}\right) - ln\left(\frac{P(c_k) = +1)}{P(c_k) = -1)}\right) \quad (2.2)$$

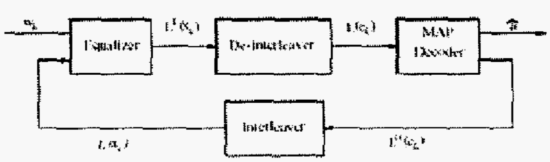

Fig. 1 Receiver model for Turbo equalizer

The extrinsic LLR's of the decoder are interleaved to obtain the intrinsic information for the equalizer i.e., $L(x_k) = \Pi(L^O(c_k))$, $k = 1, 2, \ldots, K$. The decoder also outputs estimates of the data bits as

$$a_k = \arg\max_{b\ 0,1} \{P(b_k = b / L(c_1), (c_2), \ldots, L(c_k)\} \quad 2.3$$

This process is repeated several times over a block of received symbols. The BCJR algorithm and its *M*-variant as used in Turbo equalizer are described in next paragraph.

3. SISO Module Based on the BCJR Algorithm.

In our description of the BCJR algorithm, we assume that the encoder trellis is time-invariant. This is a valid assumption because we are dealing with the timeinvariant trellises of the DTTF and convolutional codes in IED. For notation, we use the trellis section of the encoder trellis shown in Figure 2. The symbol *e* denotes a trellis edge starting from state S_e^α and ending at state S_e^β. u_e and c_e are the respective information and the code symbols associated with the edge *e*.

The SISO module can operate at *bit* level or *symbol* level. Quite often, the interleaver operates at *bit* level for improved performance. This necessitates the transformation of *symbol* LLR's into *bit* LLR's and *vice versa* if the SISO module is implemented at the *symbol* level. Here, we describe the *symbol* level SISO module.

3.1 The Input-Output Relationships of the SISO Module

Assume that the information and code symbols are defined over a finite time index set [1,2, . . . ,K]. Let the operator

$$\log_j^*(a_j) \cong \log \sum_{j=1}^{J} e^{aj} \qquad (3.1)$$

From the input LLR's, $\lambda^I k(u)$ and $\lambda^I k(c)$, $k = 1, 2, \ldots, K$, the output *extrinsic* LLR's, $\lambda^o k(u)$ and $\lambda^o k(c)$ (c), $k = 1, 2, \ldots, K$ are calculated as

$$\lambda_k^o(c) = \overset{*}{\underset{e:c_e = c}{\log}} \log\left\{\alpha_{k-1}(S_e{}^\alpha) + \lambda_k^I(c_e) + \beta_k(S_e{}^\beta)\right\} \quad (3.2)$$

$$\lambda_k^o(u) = \overset{*}{\underset{e:u_e = u}{\log}} \log\left\{\alpha_{k-1}(S_e{}^\alpha) + \lambda_k^I(c_e) + \beta_k(S_e{}^\beta)\right\} \quad (3.3)$$

The LLR's calculated according to (3.2) and (3.3) are termed *extrinsic* due to the fact that the computation of $\lambda_k^o(u)$ does not depend on the corresponding input LLR $\lambda_k^I(u)$ and so it can be considered as an update

3.2 SISO

The quantities $\alpha_k(\cdot)$ and $\beta_k(\cdot)$ in (3.2) and (3.3) are obtained through *forward* and *backward* recursions, respectively, as

$$\alpha_k = \frac{\overset{*}{\log}\left\{\alpha_{k-1}(s^s e) + \lambda_k^I(u_e) + \lambda_k^I(c_e)\right\}}{\overset{*}{\log}_s\left\{\overset{*}{\log}\left\{\alpha_{k-1}(s^s e) + \lambda_k^I(u_e) + \lambda_k^I(c_e)\right\}\right\}}, k = 1,2,\dots,K-1$$

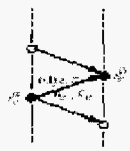

Fig. 2 Encoder trellis section defining notation for description of the SISO algorithm of the input LLR based on the code constraints and *the information provided by all homologous symbols in the sequence except the one corresponding to the same symbol interval.*

$$\beta_k(s) = \left\{\frac{\overset{*}{\log}\left\{\beta_{k+1}(s^E e)\right\} + \lambda_{k+1}^I(u_e) + \lambda_{k+1}^I(c_e)\}}{\overset{*}{\log}_s\{\alpha_k(s^E e) + \lambda_{k+1}^I(u_e) + \lambda_{k+1}^I(c_e)\}}\right\}, k \quad 1,2,\dots,K \quad 1 \quad (3.5)$$

with initial values $\alpha_o(s) = \quad 0, \quad s=S_0$

$\quad -\infty, \quad$ otherwise

$\beta_k(s) = \quad 0, \quad s=S_k$

$\quad -\infty, \quad$ otherwise

assuming that the encoder starts and ends in state 0. If the encoder ends in an unknown state, the initial values for the backward recursion are chosen as

$\beta_k(s) = \alpha_k(s)$.............$\forall s$

The denominators in (3.4) and (3.5) are normalization terms which help avoid numerical problems arising out of finite precision. The $\overset{*}{\log}$ operator may be simplified

$$\overset{*}{\log_j}(a_j = \max_j(a_j) + \delta(a_1, a_2, \dots, a_j) \quad (3.6)$$

where $(a_1, a_2, \dots, a_J)$ is a correction term that can be computed recursively using a single-entry lookup table. This simplification significantly decreases the computational complexity of the BCJR algorithm at the expense of slight performance degradation.

3.3 Inter-conversion between *symbol* and *bit* Level LLR's

Inter-conversion operations between *symbol* and *bit* Level LLR's are necessitated by the presence of a bit-interleaver. These operations assume that the bits forming a symbol are independent. Suppose $\mathbf{u} = [u1, u2, \dots, um]$ is a symbol formed by m bits. The extrinsic LLR $_j$ of the j the bit u_j within the symbol $\mathbf{u}$ is obtained as

$$\lambda_j^o(u) = \overset{*}{\log}_{u:u_j=1}\left[\overset{o}{\lambda}(u) + \lambda^I(u)\right] - \overset{*}{\log}_{u:u_j=0}\left[\overset{o}{\lambda}(u) + \lambda^I(u)\right] - \lambda_j^I(u) \quad (3.7)$$

Conversely, the extrinsic LLR of the symbol $\mathbf{u}$ is obtained from the extrinsic LLR's of its component bits u_j as

$$\lambda(u) \quad \sum_{j=1}^{m} \lambda_j u_j \quad (3.8)$$

4. *M*-BCJR Algorithm

The *M*-BCJR algorithm is a reduced-complexity variant of the BCJR algorithm and is based on the *M*-algorithm, a reduced-search trellis decoder. The reduction in complexity is achieved by retaining only the *M*-best paths in the forward recursion at each time instant. In the calculation of α_k through forward recursion on α_{k-1}, only the M largest components are used; the rest of them are set to an LLR of $-\infty$ and the corresponding states are thus declared dead. The backward recursion is executed only forward recursion. In Figure 3, we show an example of *M*-BCJR computation pattern for $M = 2$.

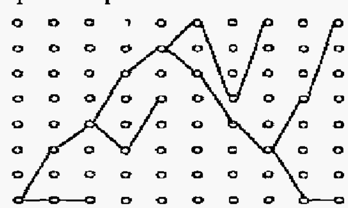

Fig. 3 Idealized computation pattern in the M-BCJR algorithm on an 8-state trellis. A line connecting two nodes indicates that the left node was used to compute the right node and that right node survived the reduction process

4.1 Performance analysis of M-BCJR Equalizer.

The performance of the *M*-BCJR equalizer is studied and contrasted with that of the BCJR equalizer on a variety of ISI channels (precoded and non-precoded) with the help of BER and FER simulations and EXIT charts. The ISI channels are modeled as convolutional codes (DTTF's). In all our simulations, the channel is assumed to be static and its coefficients f_m, $m = 0, 1, \ldots, L-1$ where L is the length of the channel impulse response, are perfectly known. Each of the channel coefficients has a power and is

$$p_m = \left|f_m^2\right|, \qquad m = 0,1,\ldots,L-1$$

normalized such that the total power

$$\sum_{m=0}^{L-1} p_m = 1$$

We shall represent an ISI channel by its coefficients $[f_0, f_1, \ldots, f_{L-1}]$. We also investigate the performance of the *M*-BCJR equalizer on precoded channels because precoding improves the asymptotic performance of SCCC's. Precoding is achieved by appropriately processing the interleaved bits stream prior to passing it through the DTTF.

4.2 Performance of the *M*-BCJR Equalizer

The BER and FER curves of the *M*=8 BCJR algorithm are almost overlapping with those of the full BCJR which operates on the whole 16-state trellis. When *M*=4 is used, the loss in performance is only 0.05 dB at a BER of 10^{-5}. For *M*=3, the loss is 0.25 dB at a BER of 10^{-4}. For *M*=2, the IED algorithm fails to evolve and does not provide any improvement in performance with iterations. As can be seen from these results for the above channel, we may use the *M*=4 BCJR equalizer with virtually no performance degradation or the *M*=3 BCJR equalizer with a very small loss in performance. This is an interesting result and suggests that the complexity of the BCJR equalizer can be reduced considerably without sacrificing its performance.

In the remainder of this section, we present a few more simulation results to show that the *M*-BCJR equalizer delivers similar performance on a majority of ISI channels, if not all of them. We present simulation results in based on EXIT charts which demonstrate that the *M*-BCJR equalizer suffers negligible losses on practically every ISI channel even for very small values of *M*. In Figure 6, we present the BER simulation results over 6 iterations for the unprecoded [0.5, 0.5, 0.5, 0.5] channel using the full BCJR and *M*=3 BCJR equalizers. An information block length of L=2048 was used. The full BCJR equalizer operates on the whole trellis consisting of 8 states at each time instant. From the plots, we observe that the performance of the *M*=3 BCJR equalizer is almost indistinguishable from that of the full BCJR equalizer in the region of BER=10^{-5} at reasonably high E_b/N_o. The performance of the *M*=3 BCJR equalizer is relatively worse at low E_b/N_o. The performance of the turbo equalizer saturates at a BER of 10^{-5} and does not improve significantly even at very high *Eb/No*. Such an early error floor is typical of SCC's in which the inner code is non-recursive . It can be avoided by precoding the channel. The BER and FER simulation results for the precoded [0.5, 0.5, 0.5, 0.5] channel using the full BCJR, M=5 and M=3 BCJR equalizers are plotted in Figure 9. In comparison with the non-precoded channel, the M-BCJR equalizer suffers significant losses in the precoded case. The performance of the M=5 BCJR equalizer is approximately 0.25 dB worse than the BCJR algorithm at a BER of 10^{-4}. However, this difference diminishes as we progress toward smaller BER's. For *M*=3 BCJR, the IED hardly yields any improvement in the BER performance with increasing number of iterations. This is in stark contrast with its performance on the non-precoded channel. It is also interesting to note that although the asymptotic performance on increasing the number of iterations is not of much help in the low regions of SNR. The precoded channels is better at high E_b/N_o, the non-precoded channels offer better performance during the first few iterations.

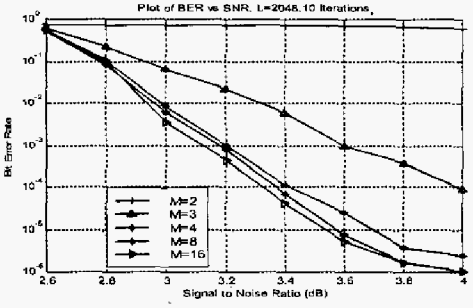

Fig4 BER & FER: Non-precoded $[\sqrt{0.45},\sqrt{0.25},\sqrt{0.15},\sqrt{0.10},\sqrt{0.05}]$, $[1, 3]_8$

This behavior is a result of the fact that the initial reliability of a precoded channel is smaller than that of a non-precoded channel. However, as the iterations outperforms the non-precoded channel.

Also, there are no signs of an error floor at a BER of 10^{-5} and thus the performance may improve significantly as E_b/N_o increases. In the middle to high regions of SNR, when the number of iterations increases from 1 to 8, the performance of the Turbo decoder improves dramatically. In other words, BER decreases dramatically. This is due to the decoder 1 and decoder 2 share the information and make more accurate decisions.

As the number of iteration increases, the performance of the Turbo decoder improves. However, after the number of iterations reaches a certain value, the improvement is not significant. It can be explained that decoder 1 and 2 already have enough information, further iterations do not give them more information. If the number iterations are increased (Eg: #18) the SNR improvement is only 0.3dB but the decoding delay is more. Hence we can use only 6 iterations.

5. Conclusion

The simulation results show that Turbo code is a powerful error correcting coding technique in low SNR environments. It has achieved near Shannon capacity. However, there are many factors to be considered in the Turbo code design. First, a trade-off between the BER and the number of iterations need to be made, e.g., more iterations lower BER is obtained, but the decoding delay is increases.

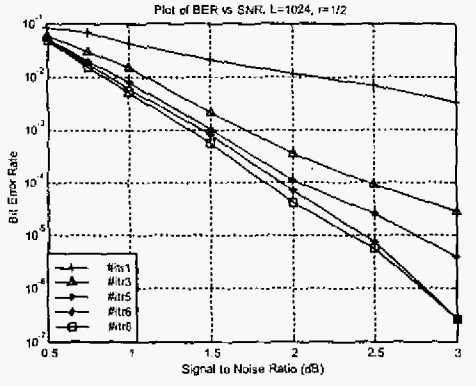

Fig.5 Effect of number of iteration on Bit Error Rate.

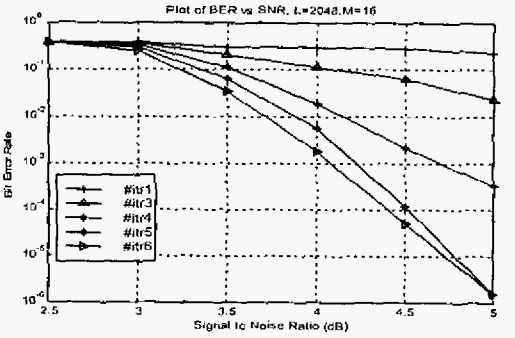

Fig. 6 Simulation results: 1.D precoded [0.5, 0.5, 0.5, 0.5], $[1, 23/35]_8$

Secondly, the effect of the frame size on the BER also needs to be considered. Although the Turbo code with larger frame size has better performance, the output delay is longer. Thirdly, the code rate is another factor that needs to be considered. The higher coding rate needs more bandwidth. From the simulation results, it is observed that the drawback of the Turbo code is its complexity and also the decoding time. Simulation results showed that the *M*-BCJR algorithm suffers significant losses in the case of simple convolutional decoders. This contrasting behavior of the *M*-BCJR algorithm in the cases of ISI channels and convolutional codes has been explained. It can be attributed to the metrics computed during the forward and backward recursions. The *M*-BCJR equalizer has much larger variance than in the case of a convolutional code. The larger variance of the metrics in the *M*-BCJR equalizer makes the algorithm less sensitive to the paths discarded.

References

[1] C. Douillard, A. Picart, M. Jézéquel, P. Didier, C. Berrou, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Commun.*, vol. 6, pp. 507–511, Sept.–Oct. 1995.

[2] M. J. Gertsman and J. L. Lodge, "Symbol-by-symbol MAP demodulation of CPM and PSK signals on Rayleigh flat-fading channels," *IEEE Trans. Commun.*, vol. 45, pp. 788–799, July 1997.

[3] G. Bauch, H. Khorram, and J. Hagenauer, "Iterative equalization and decoding in mobile Communications systems," in *Proc. the European Personal Mobile communications Conf.*, Bonn, Germany, Sept. 30–Oct. 2 1997,

[4] M. Moher, "Decoding via cross-entropy minimization," in *Proc. IEEE Global TelecommunicationsConf. 1993*, Houston, TX, Nov. 29 –Dec. 2 1993, pp. 809–813.

[5] G. Bauch and V. Franz, "Iterative equalization and decoding for the GSM-system," in *Proc. IEEE 48th Vehicular Technology Conf.*, Ottawa, Canada, May 18–21, 1998.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. Int. Conf. Communications*, Geneva, Switzerland, May 23-26, 1993 pp. 1064–1070. [7] C. Berrou and A. Glavieux, "Near optimum error-correcting coding and decoding: Turbo codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[8] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," in *Proc. Global Telecommunications Conf. 1997*, Phoenix, AZ, Nov. 3–8, 1997, pp. 639–641.

[9] T. Souvignier, A. Friedman, M.Öberg, R. E. S. P. H. Siegel, and J.Wolf, "Turbo decoding for pr4:Parallel versus serial concatenation," in *Proc. Int. Conf. communications 1999*, Vancouver, Canada, June 6–10, 1999.

[10] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[11] H. Nickl, J. Hagenauer, and F. Burkett, "Approaching Shannon's capacity limit by 0.27 dB using simple hamming codes," *IEEE Commun. Lett.* vol. 1, pp. 130–132, Sept. 1997. H. Nickl, J. Hagenauer, and F. Burkett, "Approaching Shannon's capacity limit by 0.27 dB using simple hamming codes," *IEEE Commun. Lett.* vol. 1, pp. 130–132, Sept. 1997.

[12]R.Bahl, J.Cocke, F.Jelinek, and J.Racic,"Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans, Inform. Theory, pp.284-287, 1974.

[13]S.Benedetto, D.Divsalar, G.Montorsi, and F.Pollara "A soft-input soft-output Maximum A Posteriori (MAP) module to decode parallel and serial concatenated codes", TDA progress report 42-127, November 15, 1996.

[14]Mustafa Eroz, A.Roger Hammons Jr., "On the design of prunable interleavers for Turbo codes," in Proc. IEEE VTC'99, Houston, TX, May15-19, 1999.

[15]D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding" in *Proc. IEEE GLOBECOM*, Phoenix, AZ, Nov. 1997, vol. 2, pp. 639–643.

[16]A. Anastaspoulos and K. Chugg, "Iterative equalization/decoding for TCM for frequency-selective fading channels," in *Proc. 31st Asilomar Conf. Signals, Systems & Computers*, Pacific Grove, CA, Nov. 1997, vol. 1, pp. 177–181.

[17]A. Berthet, R. Visoz and P. Tortelier, "Sub-optimal turbo-detection for coded8-PSK signals over ISI channels with applications to EDGE advanced mobile systems," in *Proc. 11th Int. Symp. Personal, Indoor and Mobile Radio Communications*, London, UK, Sept. 2000, vol. 1, pp. 151–157.

[18]S. Haykin, *Communication Systems*, 3rd ed. New York: Wiley, 1994.

[19]Heegurd and Wickun, *Turbo coding, Kluwer Academic Publisher London 1999.*

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

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